BETTER OPTIMIZATION FOR NEURAL ARCHITECTURE SEARCH WITH MIXED-LEVEL REFORMULATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Many recently proposed methods for Neural Architecture Search (NAS) can be formulated as bilevel optimization. For efficient implementation, its solution requires approximations of second-order methods. In this paper, we demonstrate that gradient errors caused by such approximation leads to suboptimality, in that such a procedure fails to converge to a (locally) optimal solution. To remedy this problem, this paper proposes MiLeNAS, a mixed-level reformulation for Neural Architecture Search that can be optimized more reliably. It is shown that even when using a simple first-order method on mixed-level formulation, MiLeNAS can achieve lower validation error for NAS problems. Consequently, architectures obtained by our method achieve consistently higher accuracies than those obtained from bilevel optimization. Moreover, the use of first-order updates in our method also leads to faster training. Extensive experiments within convolutional architecture search space validate the effectiveness of our approach.

1 INTRODUCTION

The success of deep learning in many applications heavily depends on novel neural architectures (He et al., 2016; Huang et al., 2017). However, most widely-employed architectures are developed manually, making them time-consuming and error-prone. Thus, there has been a growing interest in Neural Architecture Search (NAS), which automates the manual process of architecture design (Bello et al., 2016; Real et al., 2017). There are three major methods for NAS: evolutionary algorithms (Real et al., 2017; Elsken et al., 2018), reinforcement learning-based methods (Bello et al., 2016; Pham et al., 2018), and gradient-based methods (Liu et al., 2018b; Xie et al., 2018; Elsken et al., 2018; Luo et al., 2018). Evolutionary algorithms and reinforcement learning-based methods require thousands of GPU days to achieve state-of-the-art performance, while gradient-based methods can finish searching within as little as several GPU days.

We can formulate the gradient-based method as a bilevel optimization problem:

$$\min_{\alpha} \quad \mathcal{L}_{\text{val}}(w^*(\alpha), \alpha) \tag{1}$$

$$\text{s.t.} \quad w^*(\alpha) = \arg\min_{w} \ \mathcal{L}_{\text{tr}}(w, \alpha) \tag{2}$$

where w represents the network weight and α determines the neural architecture. $\mathcal{L}_{tr}(w, \alpha)$ and $\mathcal{L}_{val}(w, \alpha)$ denote the loss with respect to training data and validation data with w and α, respectively. Though bilevel optimization can accurately describe the NAS problem, it is difficult to solve, as obtaining $w^*(\alpha)$ in Equation 2 requires one to completely train a network for each update of α. Current methods used in NAS to solve bilevel optimization are heuristic, and $w^*(\alpha)$ in Equation 2 is not satisfied due to first-order or second-order approximation (Liu et al., 2018b; Dong & Yang, 2019). Hence, these algorithms lack theoretical guarantees and only achieve low convergence rates.

Single-level optimization is another method used to solve the NAS problem and is defined as:

$$\min_{w,\alpha} \mathcal{L}_{\text{tr}}(w, \alpha)$$

which can be solved efficiently by stochastic gradient descent. However, single-level optimization commonly leads to overfitting with respect to α, meaning that it cannot guarantee that the validation loss $\mathcal{L}_{\text{val}}(w, \alpha)$ is small. This directly contradicts the objective of NAS, which is to minimize validation loss to find optimal structures. Therefore, single-level optimization is insufficient for NAS.

In this work, we propose mixed-level optimization, which incorporates both bilevel and single-level optimization. Rather than minimizing the validation loss with respect to α with the fully trained weights $w^*(\alpha)$ as in Equation 2, or directly minimizing α over the training loss, we minimize both the training loss and validation loss with respect to α, and training loss with respect to w, simultaneously. Note that when the hyperparameter λ (Equation 4) of our mixed-level optimization is set to zero, our mixed-level optimization method degrades to single-level optimization. Alternatively, if λ approaches infinity, our method becomes the bilevel optimization. Since we mix single-level and bilevel optimization, we call our method MiLeNAS, *mixed-level optimization* based NAS.

MiLeNAS can be a universal method to search for better architectures with a faster convergence rate. First, it has a computational efficiency similar to that of single-level optimization, but without the concern of overfitting. Second, our method can fully exploit both training data and validation data to update α. Current bilevel optimization-based algorithms, first-order DARTS (Liu et al., 2018b), SNAS (Xie et al., 2018), and GDAS (Dong & Yang, 2019), only use the validation loss in updating α, ignoring the training data. Although second-order DARTS can preserve second-order gradient in both training and evaluation data by approximating $w^*(\alpha)$ and using only a single training step, such approximation does not solve the inner optimization (Equation 2) completely. Thus, gradient error caused by such approximation leads to suboptimality. Third, MiLeNAS is a *generic* framework to model the NAS and even hyperparameter tuning problem. Once the training loss $\mathcal{L}_{\mathrm{tr}}(w, \alpha)$ and validation loss $\mathcal{L}_{\mathrm{val}}(w, \alpha)$ share the same form, MiLeNAS can be applied to these problems.

Extensive experiments validate the effectiveness of MiLeNAS. We first correlate MiLeNAS with single-level and bilevel methods by comparing their respective gap between the training loss and the evaluation loss. The result shows MiLeNAS can overcome overfitting, and single-level and bilevel optimizations are special cases of MiLeNAS. Furthermore, MiLeNAS achieves a better validation accuracy with three times faster than bi-level optimization. Evaluation on searched architectures shows MiLeNAS has reached an error rate of $2.51\% \pm 0.11\%$ (best: 2.34%) on CIFAR-10, largely exceeding bilevel optimization methods (DARTS-2.76%, GDAS-2.82%). The transferability evaluation on ImageNet shows that MiLeNAS has a top-1 error rate of 24.7% and top-5 error rate of 7.6%, exceeding bilevel optimization methods by around 1% to 2%. Moreover, we demonstrate that MiLeNAS is generic by applying it to sampling-based methods, and verify that MiLeNAS can optimize performance for different model sizes, achieving better architectures than bilevel methods.

We summarize our contributions as follows:

- We propose a novel solution to the NAS problem, by reformulating it as a mixed-level optimization instead of bilevel optimization, alleviating gradient error caused by approximation in bilevel optimization. This leads to a reliable first order method as efficient as that of the single-level method.
- MiLeNAS can search for better architectures with faster convergence rates. Extensive experiments on image classification demonstrate that MiLeNAS can achieve lower validation error and enjoy a search time three times shorter than bilevel optimization.
- MiLeNAS is a generic framework for gradient-based NAS problems. Our experiments show its versatility in sampling-based methods to obtain better architectures.

The source code of MiLeNAS is available at `https://tinyurl.com/milenas-code`

2 METHOD

2.1 MIXED-LEVEL OPTIMIZATION

Our mixed-level optimization aims to overcome the difficulty of solving bilevel optimization and to fully exploit information embedded in the training data that is not used by current NAS algorithms when updating architecture parameter α (as shown in Equation 1). We propose mixed-level optimization, which gives a trade-off between single level and bilevel optimization. It maintains a simple implementation and avoids the gradient error problem caused by second-order gradient approximation. Mixed-level optimization additionally overcomes the overfitting problem of single-level optimization.

Next, we derive the mixed-level optimization from the single-level optimization, aiming to reduce α overfitting by considering both the training and validation loss. First, the single-level optimization problem is defined as:

$$\min_{w,\alpha} \mathcal{L}_{tr}(w, \alpha) :\equiv \min_{\alpha} \mathcal{L}_{tr}(w^*(\alpha), \alpha) \tag{3}$$

where $\mathcal{L}_{tr}(w, \alpha)$ denotes the loss with respect to training data. When training neural network weights w, methods such as dropout are used to avoid overfitting with respect to w. However, directly minimizing Equation 3 to obtain the optimal weight and architecture parameter will lead to overfitting with respect to α. Because α solely depends on the training data, when it is optimized, there is a disparity between $\mathcal{L}_{tr}(w, \alpha)$ and $\mathcal{L}_{val}(w, \alpha)$. Thus, the objective function defined in Equation 3 is inadequate for neural network search.

To solve the overfitting problem of α, we resort to the most popular regularization method and use $\mathcal{L}_{val}(w, \alpha)$ as the regularization term. We minimize Equation 3 subject to the constraint

$$\mathcal{L}_{\mathrm{val}}(w^*(\alpha), \alpha) \leq \mathcal{L}_{\mathrm{tr}}(w^*(\alpha), \alpha) + \delta,$$

where δ is a constant scalar. The above constraint means that the validation loss should not be much larger than the training loss. By the Lagrangian multiplier method, we minimize

$$w^*(\alpha) = \arg\min_{w} \mathcal{L}_{tr}(w, \alpha)$$

$$\min_{\alpha}(1 - \lambda')\mathcal{L}_{tr}(w^*(\alpha), \alpha) + \lambda'\mathcal{L}_{val}(w^*(\alpha), \alpha) - \lambda'\delta, \quad \text{with} \quad 0 \leq \lambda' \leq 1.$$

Because δ is a constant which does not affect the minimization, after normalizing the parameter before $\mathcal{L}_{tr}(w(\alpha), \alpha)$ to one, we obtain the following mixed-level optimization using Equation 3:

$$\min_{\alpha,w} \left[\mathcal{L}_{tr}(w, \alpha) + \lambda\mathcal{L}_{val}(w^*(\alpha), \alpha)\right], \tag{4}$$

where λ is a non-negative regularization parameter that balances the importance of training loss and validation loss. This is different from the bilevel optimization Equation 1 and 2 and single-level optimization Equation 3. Instead of only minimizing the loss with respect to α on the validation data given w as in Equation 1, the minimizer α in Equation 4 must guarantee that both $\mathcal{L}_{tr}(w, \alpha)$ and $\mathcal{L}_{val}(w, \alpha)$ are small to achieve optimality. As the name *mixed-level* implies, we mix single-level with bilevel optimization. Specifically, when $\lambda = 0$, mixed-level optimization degrades to the single level optimization, while when λ approaches infinity, mixed-level reduces to bilevel optimization. Therefore, it can be stated that both single and bilevel optimization are special cases of mixed-level optimization.

Furthermore, by taking the underlying relation between training loss and validation loss into account, our mixed-level optimization can overcome the overfitting problem and search for architectures with higher accuracy compared to single-level and bilevel optimization.

Algorithm 1 Mixed-Level Optimization based NAS.

1: **Input:** Set parameter λ (usually set to be one). Initialize w and α.
2: **while** not converge **do**
3: Update $w = w - \eta_w \nabla_w \mathcal{L}_{\mathrm{tr}}(w, \alpha)$;
4: Update the structure parameter α using the loss $\mathcal{L}_{tr}(w, \alpha) + \lambda\mathcal{L}_{val}(w, \alpha)$.
5: **end while**

Our mixed-level optimization can be solved as efficiently as the single-level method. First, we update the network weights w using only training loss. Second, we update α using both training loss and validation loss. Specifically, if we apply the mixed-level method to the gradient-based NAS problem, we only require the first-order method (stochastic gradient descent) to solve equation 4 as follows:

$$\begin{aligned} w =& w - \eta_w \nabla_w \mathcal{L}_{\mathrm{tr}}(w, \alpha) \\ \alpha =& \alpha - \eta_\alpha \left(\nabla_\alpha \mathcal{L}_{\mathrm{tr}}(w, \alpha) + \lambda \nabla_\alpha \mathcal{L}_{\mathrm{val}}(w, \alpha)\right), \end{aligned} \tag{5}$$

where η_w and η_α are step sizes related to w and α, respectively. Note that in this simplified first order method, when updating α, we replace $w^*(\alpha)$ by its approximation w, and we do not take the derivative with respect to $w^*(\alpha)$. As our experiments will demonstrate, this method can optimize the mixed-level objective function more reliably than second order methods for the bi-level formulation, due to the fact that the derivative of $w^*(\alpha)$ has significantly less impact in the mixed level formulation than that of the bi-level formulation. It is possible to incorporate second order information, but we find that this is not necessary in practical NAS applications.

2.2 COMPARISON WITH BILEVEL OPTIMIZATION

Our mixed-level optimization is a generic framework for NAS problems. We apply it to the gradient-based NAS method and compare it with DARTS, whose algorithm is derived from bilevel optimization (Liu et al., 2018b). When optimizing α, DARTS attempts to approximate the gradient $\nabla_\alpha \mathcal{L}_{\text{val}}\left(w^*(\alpha), \alpha\right)$ by

$$\nabla_\alpha \mathcal{L}_{\text{val}}\left(w^*(\alpha), \alpha\right) \approx \underbrace{\nabla_\alpha \mathcal{L}_{val}\left(w', \alpha\right)}_{g_1} - \underbrace{\xi \nabla^2_{\alpha,w} \mathcal{L}_{train}(w, \alpha) \nabla_{w'} \mathcal{L}_{val}\left(w', \alpha\right)}_{g_2}, \tag{6}$$

where $w' = w - \xi \nabla_w \mathcal{L}_{train}(w, \alpha)$ denotes the weights for a one-step forward model. For efficient implementation, DARTS tries to use the finite difference to approximate g_2 in 6 simplified as

$$\nabla_\alpha \mathcal{L}_{\text{val}}\left(w^*(\alpha), \alpha\right) \approx \nabla_\alpha \mathcal{L}_{val}\left(w', \alpha\right) - \xi \frac{\nabla_\alpha \mathcal{L}_{train}\left(w^+_{val}, \alpha\right) - \nabla_\alpha \mathcal{L}_{train}\left(w^-_{val}, \alpha\right)}{2\epsilon^{val}} \tag{7}$$

where $w^{\pm}_{val} = w \pm \epsilon^{val} \nabla_{w'} \mathcal{L}_{val}\left(w', \alpha\right)$.

However, DARTS fails to converge to a (locally) optimal solution for two reasons. First, when calculating $\nabla_\alpha \mathcal{L}_{\text{val}}\left(w^*(\alpha), \alpha\right)$, the optimal $w^*(\alpha)$ is not obtained due to one step training. Second, the finite difference approximation of g_2 causes biased gradient error. Thus, the approximation to $\nabla_\alpha \mathcal{L}_{\text{val}}$ used in DARTS leads to a serious drawback: when α is approaching to the optima, the approximate gradient related to α in equation 7 will lead to large deviation from true gradient.

In contrast, our mixed-level method only uses the first-order information (shown in Equation 5), which does not involve gradient errors caused by approximation. Stochastic gradient descent guarantees that our mixed-level optimization converges to a (locally) optimal solution. Furthermore, comparing with Equation 7 and our update of α in Equation 5, we can see that our mixed-level method requires much fewer operations for forward and backward propagation than the bilevel method (2nd-order DARTS), resulting in a faster convergence speed.

2.3 OVERALL PIPELINE

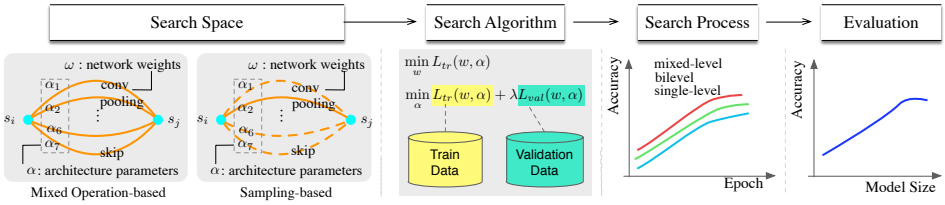


Figure 1: Overview of the pipeline designed to evaluate MiLeNAS.

As the pipeline in Figure 1 illustrated, we evaluate MiLeNAS extensively in three ways: 1. applicability in different search space or NAS methods; 2. comparison with single-level and bilevel optimization methods; 3. verification on different model sizes.

First, since our proposed MiLeNAS is a generic framework, we evaluate our method in two search space settings. The first one is the mixed-operation search space defined in DARTS, where architecture search only performs on convolutional cells to find candidate operations (e.g., convolution, max pooling, skip connection, zero) between nodes inside a cell. To make the search space continuous, we relax the categorical choice of a connection to a softmax over all possible operations:

$$\bar{o}^{(i,j)}(x) = \sum_{k=1}^{d} \underbrace{\frac{\exp(\alpha_k^{(i,j)})}{\sum_{k'=1}^{d} \exp(\alpha_{k'}^{(i,j)})}}_{p_k} o_k(x) \tag{8}$$

The weight p_k of the mixed operation $\bar{o}^{(i,j)}(x)$ for a pair of nodes (i, j) is parameterized by a vector $\alpha^{i,j}$. Thus, all architecture operation options inside a network (model) can be parameterized as α. By this definition, MiLeNAS aims at simultaneously optimizing architecture parameters α and

model weights w. Another setting is the sampling search space: instead of the mixed operation as Equation 8, GDAS uses differentiable sampler (Gumbel-Softmax) to choose an operation between two nodes in a cell. We substitute bilevel optimization in GDAS with mixed-level optimization to verify the versatility of MiLeNAS. More details of search space are covered in Appendix § A.

Second, we run different optimization methods to compare their validation accuracy, including bilevel optimization, single-level optimization, and MiLeNAS with different λ settings.

Third, the assumption that the model size determines the maximum accuracy motivates us to investigate how optimization methods perform in different model sizes. Throughout the search process, we track the best validation accuracy for every parameter number range, which is calculated by counting the number of convolutional operations in the searched cell. We did not manually define parameter number range because discrete operation choice naturally determines it.

3 Experiments and Results

In this section, we first introduce our settings and present experimental results on a comparison of MiLeNAS with single-level and bilevel methods. We then demonstrate the applicability of MiLeNAS in other NAS methods. We further evaluate the accuracy of searched architectures with different model sizes. Finally, we conclude the transferability of MiLeNAS on ImageNet.

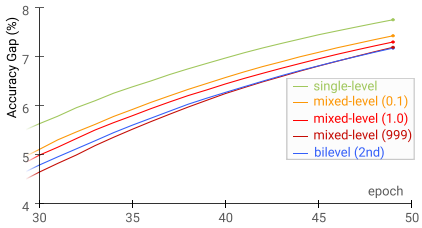
3.1 Settings

MiLeNAS contains two stages: architecture search and architecture evaluation. For the first search and evaluation stages, MiLeNAS is applied to image classification dataset CIFAR-10 (Krizhevsky et al., 2009). ImageNet was used for the transferability evaluation. To maintain a fair comparison, a search space definition similar to that of DARTS was chosen. Details regarding search space are continued in Appendix § A. The search was run under two settings: 50 epochs and 200 epochs. Each search round was run four times. In the evaluation stage, the architecture with the highest validation accuracy in the search stage was chosen with confidence (Experimental evidence has proven that the highest validation accuracy for runs in the search stage is consistent with the highest accuracy in the evaluation stage). Our code implementation is based on PyTorch 1.2.0 and Python 3.7.4. Experiments were run on NVIDIA GTX 1080Ti.

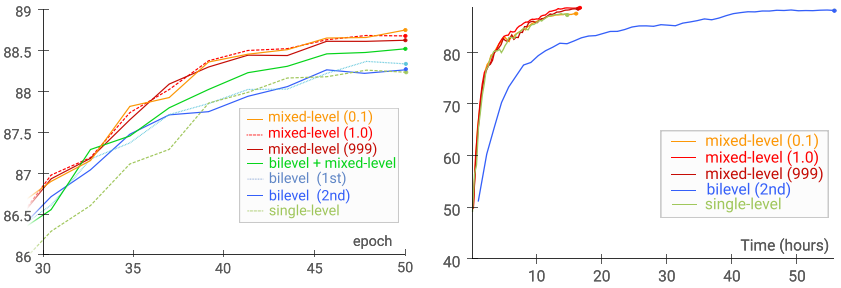
3.2 Correlation with Single-level and Bilevel methods

We correlate our mixed-level method with single-level and bilevel methods by verifying the gap between training accuracy and validation accuracy. For MiLeNAS, we choose three λ settings ($\lambda = 0.1, \lambda = 1, \lambda = 999$) to assign different proportions between training loss and validation loss. For single-level method, we update α and w both on the training dataset, while for the bilevel method, we use 2nd-order DARTS (Liu et al., 2018b). The epoch number was set to 50 (as set in second-order DARTS). In total, the five settings were run four times each, and the final result was based on the averages. From the result shown in Figure 2, we can see that the single-level method has the largest gap, while the bilevel has the smallest gap. Our mixed-level method lies between them: when λ is small (0.1), the gap is closer to that of the single-level method, while when λ is large (999), the gap is closer to that of the bilevel method. This result confirms our assertion that single-level and bilevel optimizations are special cases of mixed-level optimization with $\lambda = 0$ and $\lambda \to \infty$, respectively.

Figure 2: Training and Validation Gap (average value with smoothing)



As demonstrated in Figure 3a, the single-level method gains the lowest validation accuracy due to overfitting, and more importantly, MiLeNAS with $\lambda = 1$ and $\lambda = 0.1$ achieves higher validation accuracy than the bilevel algorithm (2nd-order DARTS, on a 50-epoch setting). This phenomenon is due to the difficulty in solving a bilevel optimization (MiLeNAS takes only about $1/3$ of the computational cost of 2nd-order DARTS). When α is close to optimal, w^* in Equation 2 should be solved



(a) Validation Accuracy

(b) Training Time

Figure 3: Comparison MiLeNAS with single-level and bilevel methods.

with high precision and $\nabla_\alpha \mathcal{L}_{\text{val}}\left(w^*(\alpha), \alpha\right)$ should be well-approximated. However, this cannot be guaranteed in 2nd-order DARTS. Thus, from Figure 3a, we can observe that in the later phase of the search (e.g., after around 35th epoch), 2nd-order DARTS barely increases its validation accuracy. Furthermore, we observe that when $\lambda = 999$, MiLeNAS has a high probability of achieving similar validation accuracy to 2nd-order bilevel optimization. This proves our claim that the bilevel optimization is the special case of mixed-level optimization with λ approaching infinity.

To remedy the approximation issue and confirm the effectiveness of MiLeNAS, we performed another experiment running bilevel optimization for the first 35 epochs, then switching to our MiLeNASmethod. When comparing its result (the green curve in Figure 3a) to that of the pure bilevel optimization (the blue curve in Figure 3a), we can see that MiLeNAS continues to improve the validation accuracy in the late phase of the search process. This observation confirms that our mixed-level algorithm mitigates the gradient approximation issue, and SGD (stochastic gradient descent) guarantees an increased validation accuracy, outperforming bilevel optimization.

Furthermore, as shown in Figure 3b, MiLeNAS is over three times faster than second-order DARTS. MiLeNAS performs a faster search due to its simple first-order algorithm, while the second-order approximation in DARTS requires more gradient computation (discussed in section 2.2).

3.3 Evaluation Results on CIFAR-10

Table 1: Comparison with state-of-the-art image classifiers on CIFAR-10

Architecture	**Test Error (%)**	**Params (M)**	**Search Cost (GPU days)**	**Search Method**
DenseNet-BC (Huang et al., 2017)	3.46	25.6	-	manual
NASNet-A + cutout (Zoph et al., 2018)	2.65	3.3	2000	RL
BlockQNN (Zhong et al., 2018)	3.54	39.8	96	RL
AmoebaNet-B + cutout (Real et al., 2019)	2.55 ± 0.05	2.8	3150	evolution
Hierarchical evolution (Liu et al., 2017)	3.75 ± 0.12	15.7	300	evolution
PNAS (Liu et al., 2018a)	3.41 ± 0.09	3.2	225	SMBO
ENAS + cutout (Pham et al., 2018)	2.89	4.6	0.5	RL
DARTS (second order) (Liu et al., 2018b)	2.76 ± 0.09	3.3	4	gradient-based
SNAS (moderate) (Xie et al., 2018)	2.85 ± 0.02	2.8	1.5	gradient-based
SNAS (aggressive) (Xie et al., 2018)	3.10 ± 0.04	2.3	1.5	gradient-based
GDAS (Dong & Yang, 2019)	2.82	2.5	0.17	gradient-based
MiLeNAS	**2.51± 0.11 (best: 2.34)**	**3.87**	0.3	gradient-based
MiLeNAS	**2.80± 0.04 (best: 2.72)**	**2.87**	0.3	gradient-based
MiLeNAS	**2.50**	**2.86**	0.3	gradient-based
MiLeNAS	**2.76**	**2.09**	0.3	gradient-based

In the evaluation stage, 20 searched cells were stacked to form a larger network; it was subsequently trained from scratch for 600 epochs with a batch size of 96 and a learning rate set to 0.03. The CIFAR-10 evaluation results are shown in Table 1 (all architectures were searched using $\lambda = 1$). The test error of our method is on par with the state-of-the-art RL-based and evolution-based NAS while

using three orders of magnitude fewer computation resources. Furthermore, our method outperforms ENAS, 2nd-order DARTS, SNAS, and GDAS with both a lower error rate and fewer parameters. We also demonstrated that our algorithm could search architectures with smaller parameters while maintaining high accuracy. Figure 4 shows examples of searched architectures.

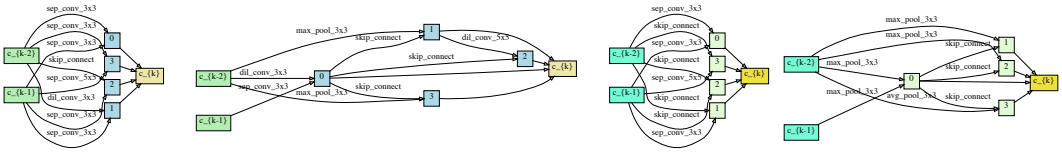


Figure 4: Searched Architectures. The left two sub-figures show an architecture that has an error rate of 2.34% with a parameter size of 3.87M; The right two sub-figures show an architecture that has an error rate of 2.50% with a parameter size of 2.86M.

Note that recently, several new NAS algorithms have been proposed which achieve lower test errors than MiLeNAS. For example, XNAS achieves a 1.81% test error (Nayman et al., 2019) and SharpDarts achieves 1.93% test error (Hundt et al., 2019). However, XNAS takes 1500 epochs and well-tuned learning rate to evaluate the chosen architecture as opposed to the 600 epochs used in both in MiLeNAS and DARTS experiments. SharpDarts defines new search spaces, which is the main reason why it obtains such a low test error.

We argue that our mixed-level optimization is a generic method that can be applied to these algorithms to improve optimization. For example, we can use the loss function defined in Equation 4 instead of Equation 1 to further improve XNAS (Nayman et al., 2019). Next, we combine mixed-level optimization with Gumbel-Softmax sampling to improve the performance of GNAS (Dong & Yang, 2019) and further demonstrate the power of mixed-level optimization.

3.3.1 Applicability to other NAS methods

MiLeNAS is universal and can substitute the bilevel optimization in other NAS methods to improve their search performance. We performed verification experiments on the Gumbel-Softmax sampling method GDAS (Dong & Yang, 2019). We reproduced GDAS[1] and substituted its bilevel optimization with MiLeNAS, denoted as MiLeNAS (Gumbel). As shown in Figure 5, MiLeNAS (Gumbel) can achieve a better validation accuracy (GDAS: 65.79%; MiLeNAS (Gumbel): *69.56%*), leading to better architectures with lower error rates (GDAS: 2.82%; MiLeNAS (Gumbel): *2.57%*).

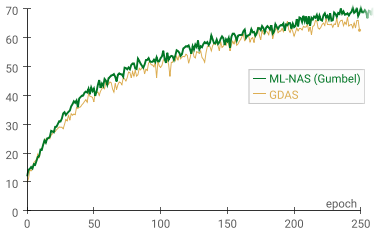


Figure 5: Applying MiLeNAS to GDAS

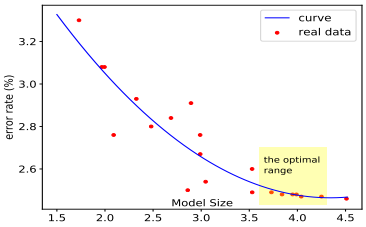


Figure 6: Evaluation on Different Model Sizes

3.3.2 Evaluation on Different Model Size

To further verify the effectiveness of MiLeNAS, we investigated the accuracy of the searched architectures in different model sizes (the parameter number of architecture). In this setting, the epoch number was set to 200 and the parameter range was tracked by using the method introduced in section 2.3. We evaluated the best architecture in each parameter range, and then found the relationship between the parameter number and the model performance (accuracy). From Figure 6, we learn that our method can also find high-performance architecture with a smaller parameter number. For example, a smaller parameter number of 2.86M gains an error rate of 2.50%, exceeding the reported

[1] As of the publication of this paper, GDAS still has not published the source code.

accuracy reported in DARTS (2.76%, 3.3M). The optimal parameter number range (between 3.5M and 4.5M) is consistent with most reported architectures in other NAS methods. Thus, evaluation on different model size verifies that MiLeNAS can solve the NAS problem better than bilevel method.

3.4 ARCHITECTURE TRANSFERABILITY EVALUATION ON IMAGENET

Table 2: Comparison with state-of-the-art image classifiers on ImageNet

Architecture	**Test Error (%)**		**Params**	$+\times$	**Search Cost**	**Search**
	top-1	top-5	**(M)**	**(M)**	**(GPU days)**	**Method**
Inception-v1 (Szegedy et al., 2015)	30.2	10.1	6.6	1448	-	manual
MobileNet (Howard et al., 2017)	29.4	10.5	4.2	569	-	manual
ShuffleNet (Zhang et al., 2018)	26.3	-	~ 5	524	-	manual
NASNet-A (Zoph et al., 2018)	26.0	8.4	5.3	564	2000	RL
AmoebaNet-A (Real et al., 2019)	25.5	8.0	5.1	555	3150	evolution
AmoebaNet-C (Real et al., 2019)	24.3	7.6	6.4	570	3150	evolution
PNAS (Liu et al., 2018a)	25.8	8.1	5.1	588	~ 225	SMBO
DARTS (Liu et al., 2018b)	26.7	8.7	4.7	574	1	gradient-based
SNAS (Xie et al., 2018)	27.3	9.2	4.2	522	1.5	gradient-based
GDAS (Dong & Yang, 2019)	27.5	9.1	4.4	497	0.17	gradient-based
GDAS (Dong & Yang, 2019)	26.0	8.5	5.3	581	0.21	gradient-based
MiLeNAS	**25.4**	**7.9**	**4.9**	**570**	**0.3**	gradient-based
MiLeNAS	**24.7**	**7.6**	**5.3**	**584**	**0.3**	gradient-based

Transferability is a crucial criterion used to evaluate the potential of the learned cells (Zoph et al., 2018). To show if the cells learned through our method on CIFAR-10 can be generalized to larger datasets, we use the same cells as in CIFAR-10 for the classification task on ImageNet. More details of the transfer learning experiments can be found in Appendix B.1.1. Table 2 presents the results of the evaluation on ImageNet and shows that the cell found by our method on CIFAR-10 can be successfully transferred to ImageNet. Our method can find smaller cell architectures that achieve relatively better performance with three times faster than the bi-level method (2nd-order DARTS).

4 RELATED WORKS

There are three major methods for Neural Architecture Search (NAS). The first method relies on evolutionary algorithms (Real et al., 2017; Elsken et al., 2018). These algorithms can simultaneously optimize architectures and network weights. However, their demand for enormous computational resources makes them highly restrictive. For example, AmoebaNet requires 3150 GPU days to achieve state-of-the-art performance (Real et al., 2019). The second method, reinforcement learning (RL) based NAS, formulates the design process of a neural network as a sequence of actions and regards the model accuracy as reward (Bello et al., 2016; Pham et al., 2018). The third method is gradient-based (Liu et al., 2018b; Xie et al., 2018; Elsken et al., 2018; Luo et al., 2018). This relaxes the categorical design choices to continuous variables and then leverages the efficient gradient back-propagation so that it can finish searching within as little as several GPU days.

5 CONCLUSION

We proposed MiLeNAS, a novel perspective to the NAS problem and reformulated it as mixed-level optimization instead of bilevel optimization. MiLeNAS can alleviate gradient error caused by approximation in bilevel optimization and enjoys the first-order efficiency as that of the single-level method. Thus, MiLeNAS can search for better architectures with faster convergence rate. Extensive experiments on image classification have demonstrated that MiLeNAS can gain lower validation error while enjoying a search time three times shorter than 2nd-order bilevel optimization. MiLeNAS is a generic method. Applicability experiments verify that it can be used to the sampling-based method to search for better architectures.

REFERENCES

Irwan Bello, Barret Zoph, Vijay Vasudevan, and Quoc V. Le. Neural optimizer search with reinforcement learning. 2016.

Xuanyi Dong and Yi Yang. Searching for a robust neural architecture in four gpu hours. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1761–1770, 2019.

Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Efficient multi-objective neural architecture search via lamarckian evolution. *arXiv preprint arXiv:1804.09081*, 2018.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

Andrew Hundt, Varun Jain, and Gregory D Hager. sharpdarts: Faster and more accurate differentiable architecture search. *arXiv preprint arXiv:1903.09900*, 2019.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 19–34, 2018a.

Hanxiao Liu, Karen Simonyan, Oriol Vinyals, Chrisantha Fernando, and Koray Kavukcuoglu. Hierarchical representations for efficient architecture search. *arXiv preprint arXiv:1711.00436*, 2017.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018b.

Renqian Luo, Fei Tian, Tao Qin, Enhong Chen, and Tie-Yan Liu. Neural architecture optimization. In *Advances in neural information processing systems*, pp. 7816–7827, 2018.

Niv Nayman, Asaf Noy, Tal Ridnik, Itamar Friedman, Rong Jin, and Lihi Zelnik-Manor. Xnas: Neural architecture search with expert advice. *arXiv preprint arXiv:1906.08031*, 2019.

Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. In *International Conference on Machine Learning*, pp. 4092–4101, 2018.

Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc V Le, and Alexey Kurakin. Large-scale evolution of image classifiers. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 2902–2911. JMLR. org, 2017.

Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 4780–4789, 2019.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. Snas: stochastic neural architecture search. *arXiv preprint arXiv:1812.09926*, 2018.

Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6848–6856, 2018.

Zhao Zhong, Zichen Yang, Boyang Deng, Junjie Yan, Wei Wu, Jing Shao, and Cheng-Lin Liu. Blockqnn: Efficient block-wise neural network architecture generation. *arXiv preprint arXiv:1808.05584*, 2018.

Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8697–8710, 2018.

A SEARCH SPACE DEFINITION

The network is formed by stacking convolutional cells for multiple times. Cell k takes the outputs of cell $k-2$ and cell $k-1$ as input. Each cell contains seven nodes: two input nodes, one output node, and the other four intermediate nodes inside the cell. The input of the first intermediate node is set equal to two input nodes, and the other intermediate nodes take all previous intermediate nodes' output as input. The output node concatenates all intermediate nodes' output in depth-wise. There are two types of cells: the normal cell and the reduction cell. The reduction cell is designed to reduce the spatial resolution of feature maps, locating at the 1/3 and 2/3 of the total depth of the network. Architecture parameters determine the discrete operation value between two nodes. All normal cells and all reduction cells share the same architecture parameters $\boldsymbol{\alpha}_{\mathrm{n}}$ and $\boldsymbol{\alpha}_{\mathrm{r}}$, respectively. By this definition, our method alternatively optimizes architecture parameters ($\boldsymbol{\alpha}_{\mathrm{n}}$, $\boldsymbol{\alpha}_{\mathrm{r}}$) and model weight parameters $\boldsymbol{w}$.

B EXPERIMENT DETAILS

B.1 EVALUATION

B.1.1 IMAGENET

The model is restricted to be less than 600M. A network of 14 cells is trained for 250 epochs with batch size 128, weight decay 3×10^{-5} and initial SGD learning rate 0.1 (decayed by a factor of 0.97 after each epoch). The training takes 3 days on a server within 8 GPU cards.