MiLeNAS: Efficient Neural Architecture Search via Mixed-Level Reformulation

Chaoyang He[1*] Haishan Ye[2*] Li Shen[3] Tong Zhang[4]
[1]University of Southern California [2]The Chinese University of Hong Kong, Shenzhen
[3]Tencent AI Lab [4]Hong Kong University of Science and Technology
chaoyang.he@usc.edu hsye_cs@outlook.com lshen.lsh@gmail.com tongzhang@tongzhang-ml.org

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 approximations lead to suboptimality, in the sense that the optimization procedure fails to converge to a (locally) optimal solution. To remedy this, this paper proposes MiLeNAS, a mixed-level reformulation for NAS that can be optimized efficiently and reliably. It is shown that even when using a simple first-order method on the mixed-level formulation, MiLeNAS can achieve a lower validation error for NAS problems. Consequently, architectures obtained by our method achieve consistently higher accuracies than those obtained from bilevel optimization. Moreover, MiLeNAS proposes a framework beyond DARTS. It is upgraded via model size-based search and early stopping strategies to complete the search process in around 5 hours. Extensive experiments within the convolutional architecture search space validate the effectiveness of our approach.

1. Introduction

The success of deep learning in computer vision heavily depends on novel neural architectures [7, 10]. However, most widely-employed architectures are developed manually, making them time-consuming and error-prone. Thus, there has been an upsurge of research interest in neural architecture search (NAS), which automates the manual process of architecture design [1, 23]. There are three major methods for NAS: evolutionary algorithms [23, 5], reinforcement learning-based methods [1, 21], and gradient-based methods [17, 28, 5, 19]. Developing optimization methods for the gradient-based NAS is promising since it achieves state-of-the-art performances on CNNs with less than one GPU day [17, 4].

Formally, gradient-based methods can be formulated as a bilevel optimization problem [17]:

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

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

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 losses 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 [17, 4]. The second-order approximation has a superposition effect in that it builds upon the one-step approximation of w, causing gradient error and deviation from the true gradient.

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

$$\min_{w,\alpha} \mathcal{L}_{\text{tr}}(w, \alpha), \tag{3}$$

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 sufficiently small. This directly contradicts the objective of NAS, which is to minimize the validation loss to find the 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 schemes. 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 the training loss with respect to w, simultaneously. Note that when the hyperparameter λ (Equation 5) of our mixed-level optimization is set to zero, our

*Equal contribution

mixed-level optimization method degrades to the single-level optimization. Alternatively, if λ approaches infinity, our method becomes the bilevel optimization. Since we mix single-level and bilevel optimizations, we call our method MiLeNAS, *Mixed-Level optimization* based NAS.

MiLeNAS can search with more stability and at faster speeds, and can find a better architecture with higher accuracy. First, it has a computational efficiency similar to that of single-level optimization, but it is able to mitigate the overfitting issue. Second, it can fully exploit both training data and validation data to update α and simultaneously avoid the gradient error caused by the approximation in the bilevel second-order methinod. Furthermore, MiLeNAS upgrades the general DARTS framework [17]. In this framework, we demonstrate its versatility in two search space settings (DARTS and GDAS [4]). Notably, this framework further introduces the model size-based search and early stopping strategies to largely accelerate the search speed (more details will be presented in Sections 3.3 and 5).

Extensive experiments validate the effectiveness of MiLeNAS. We first correlate MiLeNAS with single-level and bilevel methods by comparing their respective gaps between the training accuracy and the evaluation accuracy. The results show that MiLeNAS can overcome overfitting, and that single-level and bilevel optimizations are special cases of MiLeNAS. Furthermore, MiLeNAS achieves a better validation accuracy three times faster than bi-level optimization. Evaluations on searched architectures show that MiLeNAS reaches 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 a 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 the sampling-based search space. Finally, experiments with the model size-based and early stopping strategies introduced by the MiLeNAS framework further provide several benefits in neural architecture design and accelerate the search speed to 5 hours.

We summarize our contributions as follows:

- We propose a novel solution to the NAS problem by reformulating it as mixed-level optimization instead of bilevel optimization, alleviating the 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 a lower validation error at a search time three times shorter than that of bilevel optimization.
- MiLeNAS introduces a NAS framework beyond DARTS. This framework demonstrates that MiLeNAS is a generic framework for gradient-based NAS problems by demonstrating its versatility in sampling-based methods in obtaining better architectures.
- The MiLeNAS framework also introduces a model sized-based search strategy and an early stopping strategy to speed up the search process, and it also provides insights into neural architecture design.

We release the source code of MiLeNAS at `http://github.com/chaoyanghe/MiLeNAS`.

2. Related Works

While the deep architectures [25, 7, 10, 9] for convolutional neural networks (CNNs) are capable of tackling a wide range of visual tasks [13, 27, 18, 24], neural architecture search (NAS) has attracted widespread attention due to its advantages over manually designed architectures. There are three primary methods for NAS. The first method relies on evolutionary algorithms [23, 5, 30]. These algorithms can simultaneously optimize architectures and network weights. However, their demand for enormous computational resources makes them highly restrictive (e.g., AmoebaNet [22] requires 3150 GPU days). 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 a reward [1, 21]. The third method is gradient-based [17, 28, 5, 19, 4], which 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. Our work is related to this category, as we aim to further improve its efficiency and effectiveness.

Besides, several new NAS algorithms have been proposed to improve NAS from different perspectives. For example, task-agnostic NAS is proposed for the multi-task learning framework [6]; releasing the constraints of hand-designed heuristics [29] or alleviating the gap between the search accuracy and the evaluation accuracy are also promising directions [3, 14]. Moreover, recent proposed NAS methods achieve a higher accuracy than our method [20, 11, 2]. However, their improvements are due to novel searching spaces or searching strategies rather than a fundamental and generic optimization method.

3. Proposed Method

MiLeNAS aims to search for better architectures efficiently. In this section, we first introduce mixed-level reformulation and propose MiLeNAS first-order and second-order methods for Neural Architecture Search. We then explain the benefits of MiLeNAS through theoretical analysis,

which compares MiLeNAS with DARTS. Finally, we introduce the MiLeNAS framework and present additional benefits inspired by mixed-level optimization, including versatility in different search spaces, model size-based search, and early stopping strategy.

3.1. Mixed-Level Reformulation

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

$$\min_{w,\alpha} \mathcal{L}_{tr}(w,\alpha) :\equiv \min_{\alpha} \mathcal{L}_{tr}(w^*(\alpha),\alpha), \tag{4}$$

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 4 to obtain the optimal weight and architecture parameter may 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 4 is inadequate for neural network search.

To alleviate the overfitting problem of α, we resort to the most popular regularization method and use $\mathcal{L}_{val}(w,\alpha)$ as the regularization term. Specifically, we minimize Equation 4 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 imposes that the validation loss could not be much larger than the training loss. By the Lagrangian multiplier method, we minimize

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

Because δ is a constant which does not affect the minimization, after normalizing the parameter before $\mathcal{L}_{tr}(w(\alpha),\alpha)$ to 1, we obtain the following mixed-level optimization using Equation 4:

$$\min_{\alpha,w} \left[\mathcal{L}_{tr}(w^*(\alpha),\alpha) + \lambda\mathcal{L}_{val}(w^*(\alpha),\alpha)\right], \tag{5}$$

where λ is a non-negative regularization parameter that balances the importance of the training loss and validation loss. This is different from the bilevel optimization Equations 1 and 2 and single-level optimization in Equation 4. Therefore, by taking the underlying relation between the training loss and validation loss into account, our mixed-level optimization can alleviate the overfitting issue and search for architectures with higher accuracy than single-level and bilevel optimizations.

We then apply the first-order method (stochastic gradient descent) to solve Equation 5 as follows:

$$\begin{aligned} w &= w - \eta_w \nabla_w \mathcal{L}_{\mathrm{tr}}(w,\alpha), \\ \alpha &= \alpha - \eta_\alpha \left(\nabla_\alpha \mathcal{L}_{\mathrm{tr}}(w,\alpha) + \lambda \nabla_\alpha \mathcal{L}_{\mathrm{val}}(w,\alpha)\right), \end{aligned} \tag{6}$$

where η_w and η_α are step sizes related to w and α, respectively. Based on this MiLeNAS first-order method, we can utilize the finite approximation to derive MiLeNAS second-order method as follows:

1. $w = w - \eta_w \nabla_w L_{\mathrm{tr}}(w,\alpha)$,
2. Update α as follows:

$$\begin{aligned} \alpha = \alpha - \eta_\alpha \\ \cdot \left[\left(\nabla_\alpha \mathcal{L}_{val}\left(w',\alpha\right) - \xi \frac{\nabla_\alpha \mathcal{L}_{tr}\left(w^+_{val},\alpha\right) - \nabla_\alpha \mathcal{L}_{tr}\left(w^-_{val},\alpha\right)}{2\epsilon^{val}}\right)\right. \\ \left. +\lambda\left(\nabla_\alpha \mathcal{L}_{tr}\left(w',\alpha\right) - \xi \frac{\nabla_\alpha \mathcal{L}_{tr}\left(w^+_{tr},\alpha\right) - \nabla_\alpha \mathcal{L}_{tr}\left(w^-_{tr},\alpha\right)}{2\epsilon^{tr}}\right)\right] \end{aligned}$$

where $w' = w - \xi\nabla_w \mathcal{L}_{\mathrm{tr}}(w,\alpha)$, $w^{\pm}_{\mathrm{val}} = w \pm \epsilon^{val}\nabla_{w'}\mathcal{L}_{val}\left(w',\alpha\right)$, $w^{\pm}_{tr} = w \pm \epsilon^{tr}\nabla_{w'}\mathcal{L}_{tr}\left(w',\alpha\right)$. ϵ^{tr} and ϵ^{val} are two scalars. More details on the derivation of the MiLeNAS second-order method are placed into the Appendix.

Another benefit of mixed-level optimization is that it can embed more information. In fact, when updating α, the training loss can also effectively judge how well the neural network structure performs. Thus, it is better to fully exploit the information embedded in both the training and validation loss when updating α.

Next, we will analyze the benefits of mixed-level reformulation and conclude that the MiLeNAS-1st method is a better choice in solving the NAS problem.

3.2. Comparison between MiLeNAS and DARTS

MiLeNAS-1st v.s. DARTS-2nd As we discussed, mixed-level optimization avoids the overfitting issue and fully exploits training and validation data. Although DARTS-2nd also incorporates the training data, compared to MiLeNAS-1st, it has gradient deviation and searches inefficiently due to gradient approximation. To be more specific, when optimizing α in bilevel optimization (Equation 1), DARTS-2nd [17] approximates w with one step update: $\nabla_\alpha \mathcal{L}_{val}\left(w^*(\alpha),\alpha\right) \approx \nabla_\alpha \mathcal{L}_{val}\left(w - \xi\nabla_w \mathcal{L}_{train}(w,\alpha),\alpha\right)$, and then applies the chain rule to yield:

$$\begin{aligned} \nabla_\alpha \mathcal{L}_{\mathrm{val}}\left(w^*(\alpha),\alpha\right) \approx \underbrace{\nabla_\alpha \mathcal{L}_{val}\left(w',\alpha\right)}_{g_1} - \\ \underbrace{\xi\nabla^2_{\alpha,w}\mathcal{L}_{train}(w,\alpha)\nabla_{w'}\mathcal{L}_{val}\left(w',\alpha\right)}_{g_2}, \end{aligned} \tag{7}$$

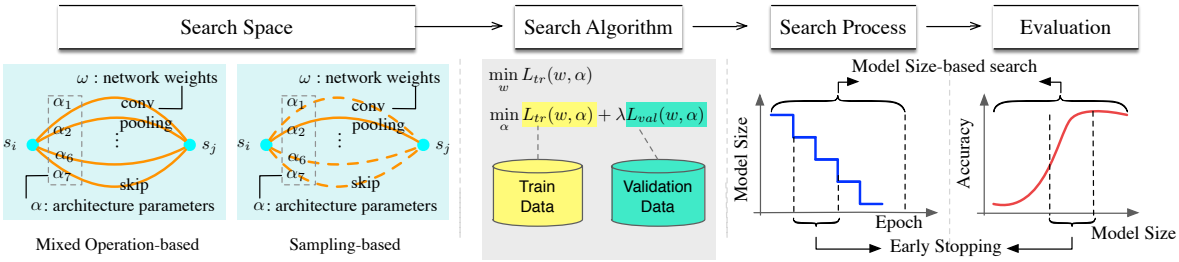


Figure 1: The Overview of the MiLeNAS Framework.

where $w' = w - \xi \nabla_w \mathcal{L}_{train}(w, \alpha)$ denotes the weights for a one-step forward model. To avoid an expensive matrix-vector product in its second term g_2, DARTS-2nd uses the finite difference approximation to reduce its complexity:

$$\nabla_\alpha \mathcal{L}_{\text{val}}\left(w^*(\alpha), \alpha\right) \approx \nabla_\alpha \mathcal{L}_{val}\left(w', \alpha\right) - \xi \frac{\nabla_\alpha \mathcal{L}_{train}\left(w_{val}^+, \alpha\right) - \nabla_\alpha \mathcal{L}_{train}\left(w_{val}^-, \alpha\right)}{2\epsilon^{val}} \quad (8)$$

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

This brings up two problems: 1. We can clearly see from Equation 8 that the second-order approximation has a superposition effect: the second-order approximation α is built upon the one-step approximation of w. This superposition effect causes gradient error, leading to deviation from the true gradient. Consequently, this gradient error may lead to an unreliable search and sub-optimal architectures; 2. Equation 8 requires two forward passes for the weights w and two backward passes for α, which is inefficient.

In contrast, our MiLeNAS-1st method only uses the first-order information (shown in Equation 6), which does not involve gradient errors caused by superposition approximation. Furthermore, comparing Equation 8 with our update of α in Equation 6, we can see that our MiLeNAS-1st requires far fewer operations, resulting in a faster convergence speed. Our experiments concur this analysis.

MiLeNAS-1st v.s. DARTS-1st DARTS also proposes to use the first-order algorithm to solve the bilevel optimization, which can be summarized as

$$w = w - \eta_w \nabla_w \mathcal{L}_{\text{tr}}(w, \alpha),$$
$$\alpha = \alpha - \eta_\alpha \nabla_\alpha \mathcal{L}_{\text{val}}(w, \alpha).$$

Although both MiLeNAS-1st and DARTS-1st share a simple form, they have fundamental differences. When updating α, MiLeNAS-1st (equation 6) takes advantage of both the training and validation losses and obtains a balance between them by setting parameter λ properly, while DARTS-1st only exploits information in validation loss $\mathcal{L}_{\text{val}}(w, \alpha)$. Therefore, MiLeNAS-1st achieves better performance than DARTS-1st. Moreover, the experiments in the original DARTS paper show that DARTS-2nd outperforms DARTS-1st since DARTS-2nd also utilizes the training loss when updating α (refer to equation 8). Thus, this also provides evidence that exploiting more information (from the training dataset) can help MiLeNAS to obtain a better performance than DARTS-1st.

MiLeNAS-1st v.s. MiLeNAS-2nd To fully understand MiLeNAS, we further investigate the effectiveness of MiLeNAS-2nd. Our experiments show that MiLeNAS-2nd is not as good as MiLeNAS-1st. Compared to MiLeNAS-1st, its searched architecture has a lower accuracy, and its search speed is slow. This conclusion supports our expectations because MiLeNAS-2nd and DARTS-2nd both have the same gradient error issue in which the second-order approximation of the true gradient causes a large deviation in the gradient descent process. This approximation only brings negative effects since MiLeNAS-1st already fully exploits the information embedded in the training and validation losses. Thus, in practice, we conclude that among these methods, MiLeNAS-1st could be the first choice in solving the NAS problem. More experimental details are covered in the appendix.

In summary, our method not only is simple and efficient, but also avoids the gradient error caused by the approximation in the bilevel second-order method. Thus, it can search with more stability and a faster speed and find a better architecture with higher accuracy.

3.3. Beyond the DARTS Framework

Motivated by the above analysis and experimental results, MiLeNAS further upgrades the general DARTS framework. As shown in Figure 1, there are three key differences.

MiLeNAS on Gradient-based Search Spaces First, since our proposed MiLeNAS is a generic framework for gradient-based NAS, we evaluate our method in two search space settings. The first 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, and 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:

$$\bar{o}^{(i,j)}(x) = \sum_{k=1}^{d} \underbrace{\frac{\exp(\alpha_k^{(i,j)})}{\sum_{k'=1}^{d} \exp(\alpha_{k'}^{(i,j)})}}_{p_k} o_k(x). \quad (9)$$

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 is the sampling search space: instead of the mixed operation as Equation 9, GDAS [4] uses a differentiable sampler (Gumbel-Softmax) to choose an operation between two nodes in a cell:

$$\tilde{p}_k^{(i,j)}(x) = \frac{\exp((\alpha_k^{(i,j)} + u_k)/\tau)}{\sum_{k'=1}^{d} \exp((\alpha_{k'}^{(i,j)} + u_k)/\tau)}, \quad (10)$$

where u_k are i.i.d samples drawn from the Gumbel$(0, 1)$ distribution and τ is the softmax temperature. We substitute bilevel optimization in GDAS with mixed-level optimization to verify the versatility of MiLeNAS.

In fact, we can design any search space using the MiLeNAS framework. In this paper, we demonstrate mixed-level optimization using DARTS and GDAS.

Model Size-based Searching We propose the model size-based searching, which is defined as searching optimal architectures in different model sizes in a single run. To be more specific, during the search, we track the model size and its best validation accuracy after every epoch, then evaluate the performance of the optimal architecture in each model size. The advantage is that we can get multiple architectures with different parameter sizes with only a single run. Our motivations are as follows: 1) to fully understand the search process with different optimization methods, we use model size-based search and find that MiLeNAS is more reliable in the search process: it stably acts in a regular model size evolution pattern (will be introduced in Section 5); 2) we hypothesize that a good NAS search method can fully exploit the accuracy in different model sizes, meaning that in the search process, the architecture with the highest validation accuracy in each model size is expected to perform excellently after architecture evaluation. This is largely ignored by previous NAS methods. In Section 5, we present experimental results of this search strategy and provide some insights for neural architecture design.

Early Stopping Strategy Early stopping strategy is motivated by the observation of the search process when using model size-based search. We find that after a certain number of epochs (around 25 epochs in DARTS and MiLeNAS), the model size will decrease. Since we know that larger model sizes may lead to better performance, we stop searching if the model size is less than the expected size. Through our experimental analysis, by drawing the relationship between the model size and the model performance (accuracy), we can determine the best stopping timing during the search process (will be introduced in Section 5).

With the improvements discussed above, we summarize the MiLeNAS framework as Algorithm 1.

Algorithm 1 MiLeNAS Algorithm

```
Define the search space;
while not converge do
  for e in epoch do
    for minibatch in training and validation data do
      w = w − η_w ∇_w L_tr(w, α);
      α = α − η_α (∇_α L_tr(w, α) + λ ∇_α L_val(w, α));
    end for
    Save the optimal structures under different model sizes;
    if current model size is less than the expected size then
      break;
    end if
  end for
end while
Evaluate on the searched neural network architecture.
```

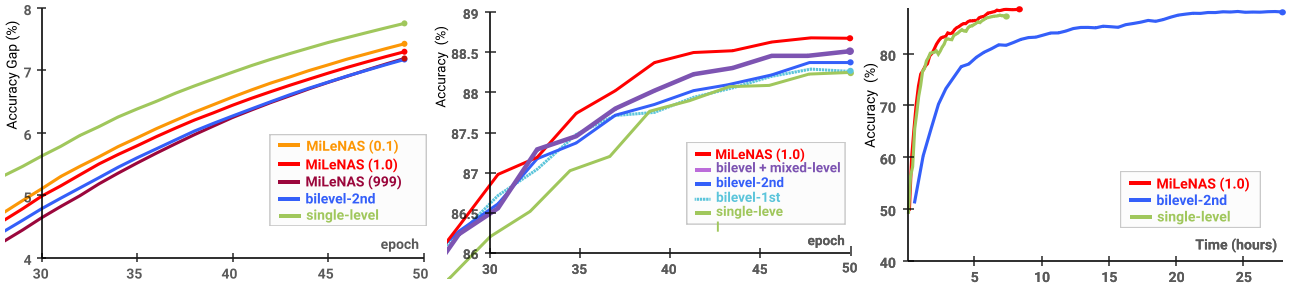
4. Experiments and Results

4.1. Settings

MiLeNAS contains two stages: architecture search and architecture evaluation. The image classification dataset CIFAR-10 [12] is used for the search and evaluation, while the ImageNet dataset is used for the transferability verification. To maintain a fair comparison, a search space definition similar to that of DARTS was chosen. In the search stage, the validation dataset is separated from the training dataset, and each method is run four times. In the evaluation stage, the architecture with the highest validation accuracy in the search stage is chosen. Our code implementation is based on PyTorch 1.2.0 and Python 3.7.4. All experiments were run on NVIDIA Tesla V100 16GB. Hyperparameter settings are kept the same as DARTS. More Details regarding the experimental settings are presented in Appendix.

4.2. Comparison with Single-level and Bilevel methods

Our intensive experimental evidence demonstrates that the architecture which has the highest validation accuracy during searching also has a larger probability to obtain the



(a) Searching Training and Validation Gap (b) Searching Validation Accuracy (c) Searching Time

Figure 2: Comparing MiLeNAS-1st with single-level and bilevel methods.

highest accuracy in the evaluation stage. Thus, to demonstrate the advantage of MiLeNAS, we first compare the validation accuracy of different methods in the search stage.

We correlate our MiLeNAS-1st method with single-level and bilevel methods by verifying the gap between training accuracy and validation accuracy to measure the overfitting with respect to the structure parameter α. For MiLeNAS, we choose three λ settings ($\lambda = 0.1, \lambda = 1, \lambda = 999$) to assign different proportions between the training loss and validation loss. For the single-level method, we update both α and w on the training dataset, while for the bilevel method, we use DARTS-2nd [17]. The epoch number is set to 50 (as set in second-order DARTS). In total, the five settings are run four times each, and the final results are based on the averages. From the results shown in Figure 2a, we see that the single-level method has the largest gap, while the bilevel method 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. Thus, 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.

As demonstrated in Figure 2b, MiLeNAS with $\lambda = 1$ achieves the highest validation accuracy. The validation accuracy of DARTS-2nd is larger than DARTS-1st (labeled as *bilevel-1st* in Figure 2b), which is the same result as in the original DARTS paper. The single-level method gains the lowest validation accuracy. This comparison of the validation accuracy is aligned with our theoretical analysis in Section 3.2: MiLeNAS is not only simple and efficient but also avoids the gradient error caused by the approximation in the bilevel second-order method.

To further confirm the effectiveness of MiLeNAS, we perform another experiment running bilevel optimization for the first 35 epochs, then switching to our MiLeNAS method. When comparing its result (the bold purple curve in Figure 2b) to that of the pure bilevel optimization (the blue curve in Figure 2b), we 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 can mitigate the gradient approximation issue, outperforming bilevel optimization.

Furthermore, as shown in Figure 2c, MiLeNAS is over three times faster than DARTS-2nd. 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 3.2).

4.3. Evaluation Results on CIFAR-10

In the evaluation stage, 20 searched cells are stacked to form a larger network, which is subsequently trained from scratch for 600 epochs with a batch size of 96 and a learning rate set to 0.025. For fair comparison, every architecture shares the same hyperparameters as the DARTS bilevel method. The CIFAR-10 evaluation results are shown in Table 1 (all architectures are 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, DARTS-2nd, SNAS, and GDAS with both a lower error rate and fewer parameters. We also demonstrate that our algorithm can search architectures with fewer parameters while maintaining high accuracy.

4.4. Transferability on ImageNet

Transferability is a crucial criterion used to evaluate the potential of the learned cells [33]. 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. Table 2 presents the results of the evaluation on ImageNet and shows that the cells found by our method on CIFAR-10 can be successfully transferred to ImageNet. Our method can find smaller cell architectures that achieve a relatively better performance at speeds three times faster than the bi-level method (DARTS-2nd). Hyperparameter Settings are presented in Appendix.

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 [10]	3.46	25.6	-	manual
NASNet-A + cutout [33]	2.65	3.3	2000	RL
BlockQNN [32]	3.54	39.8	96	RL
AmoebaNet-B + cutout [22]	2.55 ± 0.05	2.8	3150	evolution
Hierarchical evolution [16]	3.75 ± 0.12	15.7	300	evolution
PNAS [15]	3.41 ± 0.09	3.2	225	SMBO
ENAS + cutout [21]†	2.89	4.6	0.5	RL
DARTS (second order) [17]	2.76 ± 0.09	3.3	1	gradient-based
SNAS (moderate) [28]	2.85 ± 0.02	2.8	1.5	gradient-based
SNAS (aggressive) [28]	3.10 ± 0.04	2.3	1.5	gradient-based
GDAS [4]	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

* We get multiple results by using model size-based searching (introduced in Section 5); the search time is calculated without the early stopping strategy (around 8 hours). If the early stopping strategy is used, the search cost can further be reduced to around 5 hours.

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

Architecture	Test Error (%)		Params (M)	$+\times$ (M)	Search Cost (GPU days)	Search Method
	top-1	top-5				
Inception-v1 [26]	30.2	10.1	6.6	1448	-	manual
MobileNet [8]	29.4	10.5	4.2	569	-	manual
ShuffleNet [31]	26.3	-	~ 5	524	-	manual
NASNet-A [33]	26.0	8.4	5.3	564	2000	RL
AmoebaNet-A [22]	25.5	8.0	5.1	555	3150	evolution
AmoebaNet-C [22]	24.3	7.6	6.4	570	3150	evolution
PNAS [15]	25.8	8.1	5.1	588	~ 225	SMBO
DARTS [17]	26.7	8.7	4.7	574	1	gradient-based
SNAS [28]	27.3	9.2	4.2	522	1.5	gradient-based
GDAS [4]	27.5	9.1	4.4	497	0.17	gradient-based
GDAS [4]	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

* We gain multiple architectures by using model size-based searching (introduced in Section 5), and then do transfer learning on ImageNet.

5. Beyond the DARTS Framework

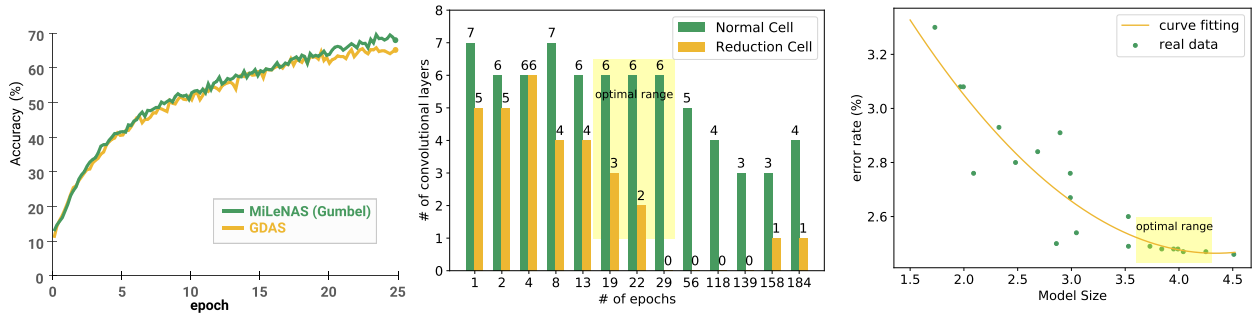
In this section, we demonstrate the effectiveness of MiLeNAS in other NAS frameworks, and then propose two strategies: model sized-based searching and early stopping.

MiLeNAS is universal and can be used as a substitute for the bilevel optimization in other NAS methods to improve their search performances. We perform verification experiments on the Gumbel-Softmax sampling method GDAS [4]. We reproduce GDAS[1] and substitute its bilevel optimization with MiLeNAS, denoted as MiLeNAS (Gumbel). As shown in Figure 3a, 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%*).

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

5.1. Model Size-based Searching

Model Size Tracking. To understand the model size evolution during the searching process, for the architecture searched in each epoch, we track the best validation accuracy for different model sizes, which is calculated by counting the number of convolution operations in the searched cell. We track the model size in this way because different discrete operation choices (determined by α) in a cell determine the model size (e.g., the model size of an architecture



(a) Applying MiLeNAS on GDAS (b) Model Size-based Searching Process (c) Error Rate over Varying Model Size

Figure 3: Evaluation on the Searched Architecture with Different Model Sizes

with more convection operations is larger than an architecture with more skip-connection operations).

Observations. By tracking the model size, we find that MiLeNAS has an obvious phase-by-phase optimization characteristic during the search process. As illustrated in Figure 3b, each phase optimizes the network architecture under a certain range of the model size (counted by the convolutional layer number) and then reduces the model size before entering another optimization phase. We evaluate the optimal architecture in each phase and find the relationship between the model size and the model performance (accuracy). From Figure 3c, we learn that when the model size increases, the model performance also increases. However, this growth reaches its limit between the optimal range (between 3.5M and 4.5M). Subsequently, the model performance is unable to improve even with an increase in the parameter number.

Our experiment on DARTS does not consistently show the same model size decreasing characteristics as in MiLeNAS. In other words, to summarize the model size and accuracy relation, we must run much more search rounds in DARTS since it does not have a stable pattern. We argue that this regular pattern seen in MiLeNAS is attributed to our mixed-level optimization since it does not suffer from gradient error by using second-order approximation.

Insight. The above observation during the search process drives our search strategy design. We define the model size-based searching as completing the search process in a large number of epochs with model size tracking, and then evaluating the network architecture accuracy under different model sizes. This provides three potential benefits for neural architecture design: 1) for a specific learning task, the most economical neural network architecture must be examined, and the redundant parameter quantity cannot bring additional benefits; 2) this method has the potential to become an alternative method for model compression since it can find multiple optimal architectures under different computational complexities; 3) most importantly, we may figure out a regular pattern between the parameter number and the architecture accuracy, allowing us to refine a strategy further to expedite the searching process. In our case, we have found that the early stopping strategy can remarkably accelerate the search speed.

5.2. Early Stopping Strategy

The timing of stopping the search is inspired by the fact that the optimal range of parameter numbers in Figure 3c (highlighted by yellow square) is found in the early phase of the searching process in Figure 3b (highlighted by yellow square). For example, we stop searching when the parameter number reaches 6 convolution operations in the normal cell and 0 convolution operation in the reduction cell. When utilizing this stopping strategy, searching for the optimal architecture on CIFAR-10 with MiLeNAS only costs around 5 hours.

6. 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 benefits from the first-order efficiency seen in single-level methods. Thus, MiLeNAS can search for better architectures with a faster convergence rate. The extensive experiments on image classification have demonstrated that MiLeNAS can gain a lower validation error at a search time three times shorter than 2nd-order bilevel optimization. MiLeNAS is a generic method. Its applicability experiments verify that it can be used in the sampling-based method to search for better architectures. Model size-based search and early stopping strategies further speed up the searching process and additionally provide several insights into neural architecture design as well.

References

[1] Irwan Bello, Barret Zoph, Vijay Vasudevan, and Quoc V. Le. Neural optimizer search with reinforcement learning. 2016. 1, 2

[2] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. *arXiv preprint arXiv:1812.00332*, 2018. 2

[3] Xin Chen, Lingxi Xie, Jun Wu, and Qi Tian. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. *arXiv preprint arXiv:1904.12760*, 2019. 2

[4] 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*, pages 1761–1770, 2019. 1, 2, 5, 7

[5] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Efficient multi-objective neural architecture search via lamarckian evolution. *arXiv preprint arXiv:1804.09081*, 2018. 1, 2

[6] Yuan Gao, Haoping Bai, Zequn Jie, Jiayi Ma, Kui Jia, and Wei Liu. Mtl-nas: Task-agnostic neural architecture search towards general-purpose multi-task learning. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[7] 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*, pages 770–778, 2016. 1, 2

[8] 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. 7

[9] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018. 2

[10] 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*, pages 4700–4708, 2017. 1, 2, 7

[11] Andrew Hundt, Varun Jain, and Gregory D Hager. sharpdarts: Faster and more accurate differentiable architecture search. *arXiv preprint arXiv:1903.09900*, 2019. 2

[12] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009. 5

[13] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 2

[14] Guohao Li, Guocheng Qian, Itzel C Delgadillo, Matthias Müller, Ali Thabet, and Bernard Ghanem. Sgas: Sequential greedy architecture search. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[15] 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)*, pages 19–34, 2018. 7

[16] Hanxiao Liu, Karen Simonyan, Oriol Vinyals, Chrisantha Fernando, and Koray Kavukcuoglu. Hierarchical representations for efficient architecture search. *arXiv preprint arXiv:1711.00436*, 2017. 7

[17] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018. 1, 2, 3, 6, 7

[18] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015. 2

[19] Renqian Luo, Fei Tian, Tao Qin, Enhong Chen, and Tie-Yan Liu. Neural architecture optimization. In *Advances in neural information processing systems*, pages 7816–7827, 2018. 1, 2

[20] 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. 2

[21] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. In *International Conference on Machine Learning*, pages 4092–4101, 2018. 1, 2, 7

[22] 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, pages 4780–4789, 2019. 2, 7

[23] 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*, pages 2902–2911. JMLR. org, 2017. 1, 2

[24] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 2

[25] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 2

[26] 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*, pages 1–9, 2015. 7

[27] Alexander Toshev and Christian Szegedy. Deeppose: Human pose estimation via deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1653–1660, 2014. 2

[28] Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. Snas: stochastic neural architecture search. *arXiv preprint arXiv:1812.09926*, 2018. 1, 2, 7

[29] Shen Yan, Biyi Fang, Faen Zhang, Yu Zheng, Xiao Zeng, Hui Xu, and Mi Zhang. Hm-nas: Efficient neural architecture search via hierarchical masking. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2019. 2

[30] Zhaohui Yang, Yunhe Wang, Xinghao Chen, Boxin Shi, Chao Xu, Chunjing Xu, Qi Tian, and Chang Xu. Cars: Continuous evolution for efficient neural architecture search. *arXiv preprint arXiv:1909.04977*, 2019. 2

[31] 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*, pages 6848–6856, 2018. 7

[32] 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. 7

[33] 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*, pages 8697–8710, 2018. 6, 7

A. Experiment Details

A.1. Search Space Definition

We adopt the following 8 operations in our CIFAR-10 experiments: 3×3 and 5×5 separable convolutions, 3×3 and 5×5 dilated separable convolutions, 3×3 max pooling, 3×3 average pooling, identity, and zero.

The network is formed by stacking convolutional cells multiple times. Cell k takes the outputs of cell $k-2$ and cell $k-1$ as its 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' outputs 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 $\boldsymbol{w}$.

A.2. Transferability on ImageNet

The model is restricted to be less than 600M. A network of 14 cells is trained for 250 epochs with a batch size of 128, weight decay 3×10^{-5}, and an initial SGD learning rate of 0.1 (decayed by a factor of 0.97 after each epoch). The training takes around three days on a server within 8 NVIDIA Tesla V100 GPU cards.

A.3. Searched Architecture

Examples of the searched architectures are shown in Figure 5.

B. Derivation of the MiLeNAS Second-Order Method

In this section, we can derive a second-order method for MiLeNAS. As in DARTS, we also approximate w^* by adapting w using only a single training step:

$$\nabla_\alpha \mathcal{L}_{\mathrm{val}}\left(w^*(\alpha), \alpha\right) \approx \nabla_\alpha \mathcal{L}_{\mathrm{val}}\left(w - \xi \nabla_w \mathcal{L}_{\mathrm{tr}}(w, \alpha), \alpha\right).$$

When applying the chain rule, we get

$$\begin{aligned}\nabla_\alpha \mathcal{L}_{\mathrm{val}}\left(w^*(\alpha), \alpha\right) \approx & \nabla_\alpha \mathcal{L}_{val}\left(w', \alpha\right) \\ & - \xi \nabla^2_{\alpha,w} \mathcal{L}_{tr}(w, \alpha) \nabla_{w'} \mathcal{L}_{val}\left(w', \alpha\right),\end{aligned}$$

where $w' = w - \xi \nabla_w \mathcal{L}_{train}(w, \alpha)$ denotes the weights for a one-step forward model. Using the finite difference approximation, the complexity of the second order derivative in Equation 7 can be simplified. If we let ϵ^{val} be a small scalar and $w^{\pm}_{val} = w \pm \epsilon^{val} \nabla_{w'} \mathcal{L}_{val}\left(w', \alpha\right)$, then:

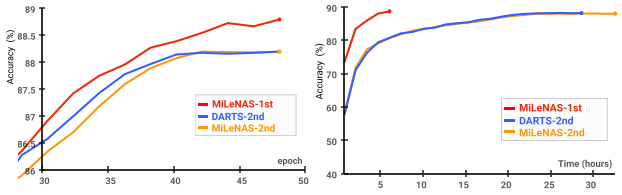
$$\begin{aligned}\nabla_\alpha \mathcal{L}_{\mathrm{val}}\left(w^*(\alpha), \alpha\right) \approx & \nabla_\alpha \mathcal{L}_{val}\left(w', \alpha\right) \\ & - \xi \frac{\nabla_\alpha \mathcal{L}_{tr}\left(w^+_{val}, \alpha\right) - \nabla_\alpha \mathcal{L}_{tr}\left(w^-_{val}, \alpha\right)}{2\epsilon^{val}}.\end{aligned}$$

Following a similar derivation of $\nabla_\alpha \mathcal{L}_{\mathrm{val}}\left(w^*(\alpha), \alpha\right)$, we have

$$\begin{aligned}\alpha = & \alpha - \eta_\alpha \\ & \cdot \left[\left(\nabla_\alpha \mathcal{L}_{val}\left(w', \alpha\right) - \xi \frac{\nabla_\alpha \mathcal{L}_{tr}\left(w^+_{val}, \alpha\right) - \nabla_\alpha \mathcal{L}_{tr}\left(w^-_{val}, \alpha\right)}{2\epsilon^{val}} \right) \right. \\ & \left. + \lambda \left(\nabla_\alpha \mathcal{L}_{tr}\left(w', \alpha\right) - \xi \frac{\nabla_\alpha \mathcal{L}_{tr}\left(w^+_{tr}, \alpha\right) - \nabla_\alpha \mathcal{L}_{tr}\left(w^-_{tr}, \alpha\right)}{2\epsilon^{tr}} \right) \right],\end{aligned}$$

where $w' = w - \xi \nabla_w \mathcal{L}_{\mathrm{tr}}(w, \alpha)$, $w^{\pm}_{\mathrm{val}} = w \pm \epsilon^{val} \nabla_{w'} \mathcal{L}_{val}\left(w', \alpha\right)$, $w^{\pm}_{tr} = w \pm \epsilon^{tr} \nabla_{w'} \mathcal{L}_{tr}\left(w', \alpha\right)$. ϵ^{tr} and ϵ^{val} are two scalars.

C. Evaluation on MiLeNAS-2nd



(a) Validation Accuracy (b) Searching Time

Figure 4: Comparison between MiLeNAS-2nd with MiLeNAS-1st.

As seen in our analysis, MiLeNAS-2nd shows a similar gradient error and is also inefficient. To confirm this, we run experiments to compare its validation accuracy and training time with MiLeNAS-1st. Each method is run four times, and the results are based on averages (Figure 4). The accuracy of MiLeNAS-2nd is lower than that of MiLeNAS-1st and similar to DARTS-2nd. The searching time of MiLeNAS-2nd is the longest because it has one more inefficient term in the second-order approximation equation. Notably, in the early phase, MiLeNAS-2nd is significantly less accurate than DARTS-2nd, which may be caused by the fact that MiLeNAS-2nd has one more term with gradient error (refer to MiLeNAS-2nd equation in Section 3.1). Thus, among these methods, MiLeNAS-1st is shown to be the optimal choice for addressing the NAS problem.

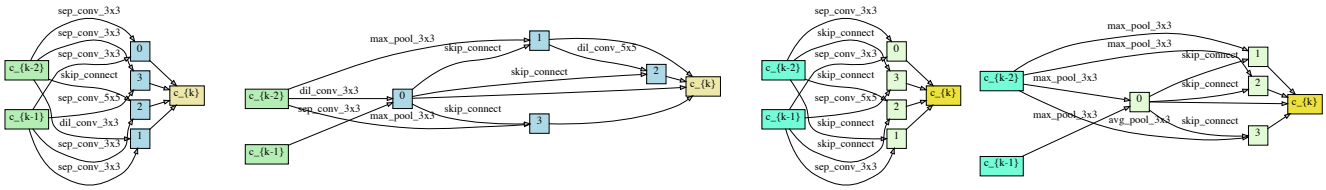


Figure 5: Searched Architectures. The left two sub-figures show an architecture that has an error rate of 2.34% with a parameter size of 3.87M; The right two sub-figures show an architecture that has an error rate of 2.50% with a parameter size of 2.86M.