Supplementary Material of MiLeNAS: Efficient Neural Architecture Search via Mixed-Level Reformulation

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 $\boldsymbol{\alpha}_{\text{n}}$ and $\boldsymbol{\alpha}_{\text{r}}$, respectively. By this definition, our method alternatively optimizes architecture parameters ($\boldsymbol{\alpha}_{\text{n}}$, $\boldsymbol{\alpha}_{\text{r}}$) 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 1.

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:

$$\nabla_{\alpha \text{val}}\left(w^*(\alpha), \alpha\right) \approx \nabla_{\alpha \text{val}}\left(w - \xi \nabla_{w \text{tr}}(w, \alpha), \alpha\right).$$

When applying the chain rule, we get

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

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:

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

Following a similar derivation of $\nabla_{\alpha \text{val}}\left(w^*(\alpha), \alpha\right)$, 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 $w' = w - \xi \nabla_w \mathcal{L}_{\text{tr}}(w, \alpha)$, $w^{\pm}_{\text{val}} = w \pm \epsilon^{val} \nabla_{w'} \mathcal{L}_{val}\left(w', \alpha\right)$, $w^{\pm}_{tr} = w \pm \epsilon^{tr} \nabla_{w'} \mathcal{L}_{tr}\left(w', \alpha\right)$. ϵ^{tr} and ϵ^{val} are two scalars.

C. Evaluation on -2nd

As seen in our analysis, -2nd shows a similar gradient error and is also inefficient. To confirm this, we run experiments to compare its validation accuracy and training time

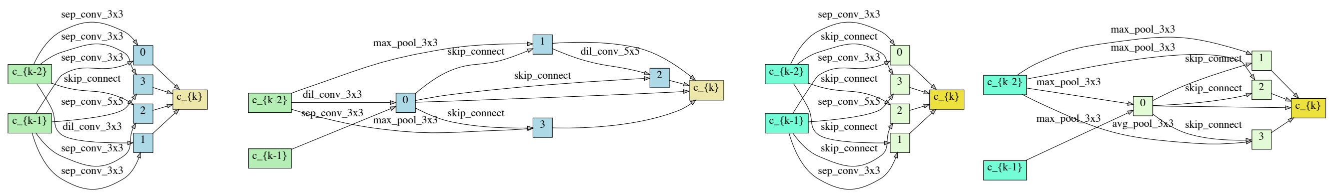
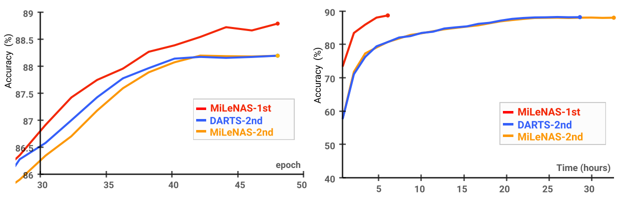


Figure 1: Searched Architectures. The left two sub-figures show an architecture that has an error rate of 2.34% with a parameter size of 3.87M; The right two sub-figures show an architecture that has an error rate of 2.50% with a parameter size of 2.86M.



(a) Validation Accuracy (b) Searching Time

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

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