## A. Weight-sharing NAS

Most RL-based NAS (e.g., Zoph & Le, 2017) and differentiable NAS (Liu et al., 2019; Cai et al., 2019b) consist of the following two stages as shown in Figure 7:

1) Stage 1 (architecture searching) - search potential architectures following a single resource constraint by using black-box optimization techniques (e.g., Zoph & Le, 2017) or differentiable weight-sharing based approaches (e.g., Liu et al., 2019; Cai et al., 2019b);
2) Stage 2 (retraining) - retrain deep neural networks (DNNs) found in step 1) from scratch for best accuracy and final deployment.

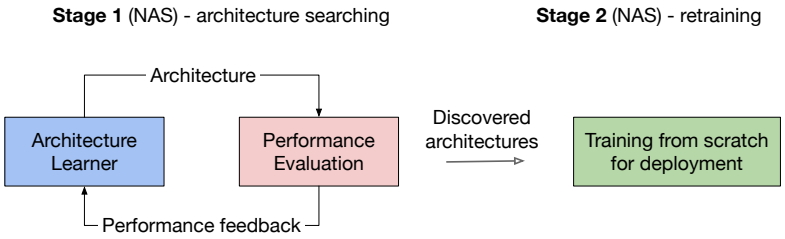


*Figure 7.* An overview of convectional NAS pipeline.

Though promising results have been demonstrated, these NAS methods usually suffer from the following disadvantages: 1) need to re-do the NAS search for different hardware resource constraints; 2) require training the selected candidate from scratch to achieve desirable accuracy; 3) 1) especially for RL-based NAS that uses black-box optimization techniques, it requires training a large number of neural networks from scratch or on proxy tasks; These disadvantages significantly increase the computational cost of NAS and make the NAS search computationally expensive.

**Supernet-based Weight-sharing NAS** To alleviate the aforementioned issues, supernet-based weight-sharing NAS transforms the previous NAS training and search procedures as follows; see Figure 8.

1) Stage 1 (supernet pretraining): jointly optimize the supernet and all possible sub-networks specified in the search space, such that all searchable networks simultaneously achieve good performance at the end of the training phase.
2) Stage 2 (searching & deployment): After stage 1 training, all the sub-networks are optimized simultaneously. One could then use typical searching algorithms, like evolutionary algorithms, to search the best model of interest. The model weights of each sub-network are directly inherited from the pre-trained supernet without any further re-training or fine-tuning.

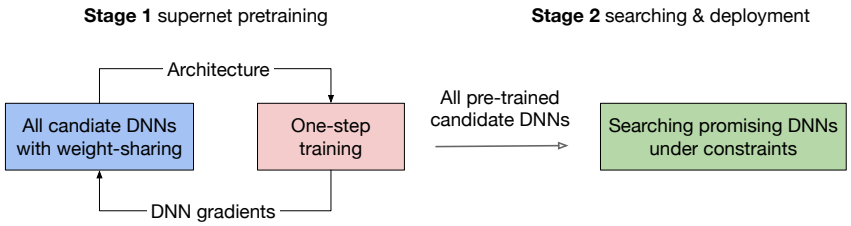


*Figure 8.* An overview of supernet-based weight-sharing NAS.

Compared to RL-based NAS and differentiable NAS algorithms, the key advantages of the supernet-based weight-sharing NAS pipeline are: 1) one needs to only perform the computationally expensive supernet training for once. All sub-networks defined in the search space are ready to use after stage 1 is fully optimized. No retraining or fine-tuning is required; 2) all sub-networks of various model sizes are jointly optimized in stage 1, finding a set of Pareto optimal models that naturally supports various resource considerations.

Notable examples of supernet-based weights-sharing NAS include BigNAS (Yu et al., 2020), OFA (Cai et al., 2019a), AttentiveNAS (Wang et al., 2020a) and HAT (Wang et al., 2020b).

## B. Weights-sharing NAS training settings

We exactly follow the training settings in Wang et al. (2020a) [2]. Specifically, we train our supernets for 360 epochs with cosine learning rate decay. We adopt SGD training on 64 GPUs. The mini-batch size is 32 per GPU. We use momeutm of 0.9, weight decay of $10^{-5}$, dropout of $0.2$, stochastic layer dropout of $0.2$. The base learning rate is set as $0.1$ and is linearly scaled up for every 256 training samples. We use AutoAugment (Cubuk et al., 2018) for data augmentation and set label smoothing coefficient to $0.1$.

We use the same search space provided in Wang et al. (2020a), see Table 7. Here Conv denotes regular convolutional layers and MBConv refers to inverted residual block proposed by Sandler et al. (2018). We use swish activation. Channel width represents the number of output channels of the block. MBPool denotes the efficient last stage in Howard et al. (2019). SE represents the squeeze and excite layer (Hu et al., 2018). *Input resolution* denotes the candidate resolutions. To simplify the data loading procedure, we always pre-fetch training patches of a fixed size, e.g., 224x224 on ImageNet, and then rescale them to our target resolution with bicubic interpolation following (Yu et al., 2020).

| Block name | Channel width | Depth | Kernel size | Expansion ratio | SE |
|---|---|---|---|---|---|
| Conv | {16, 24} | - | 3 | - | - |
| MBConv-1 | {16, 24} | {1,2} | {3, 5} | 1 | N |
| MBConv-2 | {24, 32} | {3, 4, 5} | {3, 5} | {4, 5, 6} | N |
| MBConv-3 | {32, 40} | {3, 4, 5, 6} | {3, 5} | {4, 5, 6} | Y |
| MBConv-4 | {64, 72} | {3, 4, 5, 6} | {3, 5} | {4, 5, 6} | N |
| MBConv-5 | {112,128} | {3, 4, 5, 6, 7, 8} | {3, 5} | {4, 5, 6} | Y |
| MBConv-6 | {192, 200, 208, 216} | {3, 4, 5, 6, 7, 8} | {3, 5} | 6 | Y |
| MBConv-7 | {216, 224} | {1, 2} | {3, 5} | 6 | Y |
| MBPool | {1792, 1984} | - | 1 | 6 | - |
| Input resolution | {192, 224, 256, 288} | | | | |

*Table 7.* An illustration of our search space. Every row denotes a block group.

## C. Knowledge distillation

Consider the image classification task over a set of classes $[m] := \{1, \cdots, m\}$, where we have a collection of training images and one-hot labels $\mathcal{D}^{train} = \{(x, y)\}$ with $(x, y) \in \mathcal{X} \times \mathcal{Y}$ and $y \in \{0, 1\}^m$. We are interested in designing a deep neural network $q(x; \theta) : \mathcal{X} \to \mathcal{Y}$ that captures the relationship between $x$ and $y$. Here $\theta$ is the network parameters of interest.

KD provides an effective way to train $q$ by distilling knowledge from a teacher model in addition to the one-hot labels. The teacher network is often a relative larger network with better performance. Specifically, let $p$ be the teacher network, KD enforces $q$ to mimic the output of $p$ by minimizing the closeness between $q$ and $p$, which is often specified by the KL divergence $\mathrm{D}_{\mathrm{KL}}(p \parallel q)$, yielding the following loss function,

$$\begin{aligned}
\mathcal{L}(\theta) &= (1-\beta)\mathcal{L}_{\mathrm{ERM}}(\theta) + \beta\mathcal{L}_{\mathrm{KD}}(\theta), \quad \text{with} \\
\mathcal{L}_{\mathrm{ERM}}(\theta) &= \mathbb{E}_{(x,y)\sim\mathcal{D}^{train}}\Big[\mathcal{L}(y, q(x;\theta))\Big], \\
\mathcal{L}_{\mathrm{KD}}(\theta) &= \mathbb{E}_{x\sim\mathcal{D}^{train}}\Big[\mathrm{D}_{\mathrm{KL}}(p(x) \parallel q(x;\theta))\Big].
\end{aligned} \tag{7}$$

Here $\mathcal{L}(\cdot)$ represents the empirical loss, e.g., the typical cross entropy loss $\mathcal{L}(y, q(x;\theta)) = \sum_{i=1}^{m} -y_i \log q_i$ with $q_i$ be the $i$-class probability produced by $q$. And $\mathrm{D}_{\mathrm{KL}}(p \parallel q) = \mathbb{E}_p[\log(p/q)]$. Furthermore, $\beta \in [0, 1]$ is the distilling weight that balances of the empirical loss and KD loss.

[2] https://github.com/facebookresearch/AttentiveNAS

One could also apply a temperature $T$ to soften (or sharpen) the outputs the teacher model and the student model in KD. More precisely, given an input $x$, we assume $z_i^p(x)$ and $z_i^q(x)$ the logit for the $i$-th class produced by $p$ and $q$, respectively. Then the corresponding predictions of $p$ and $q$ after temperature scaling are as follows,

$$p_i(x;T) = \mathrm{softmax}(z_i^p;T), \;\; q_i(x;T) = \mathrm{softmax}(z_i^q;T),$$

with $\mathrm{softmax}(z_i;T) = \exp(z_i/T)/\sum_i \exp(z_i/T)$. In this way, the previous KD objective (7) is now adapted to the following,

$$\begin{aligned} \mathcal{L}(\theta;T) &= (1-\beta)\mathcal{L}_{\mathrm{ERM}}(\theta) + \beta T^2 \mathcal{L}_{\mathrm{KD}}(\theta;T), \;\; \text{with} \\ \mathcal{L}_{\mathrm{KD}}(\theta;T) &= \mathbb{E}_x \Big[ D_{\mathrm{KL}}(p(x;T) \parallel q(x;T,\theta)) \Big]. \end{aligned} \tag{8}$$

Here $T^2$ is introduced to ensure the gradients from the KD loss is at the same scale w.r.t the gradients from the empirical loss, see (e.g., Hinton et al., 2015). We set $\beta = 0.9$ as default.

## D. Additional results on ablation studies

Following the settings in section 4.2, we provide further analyses on the performance of sub-networks learned under different $\alpha$ and $\beta$ settings.

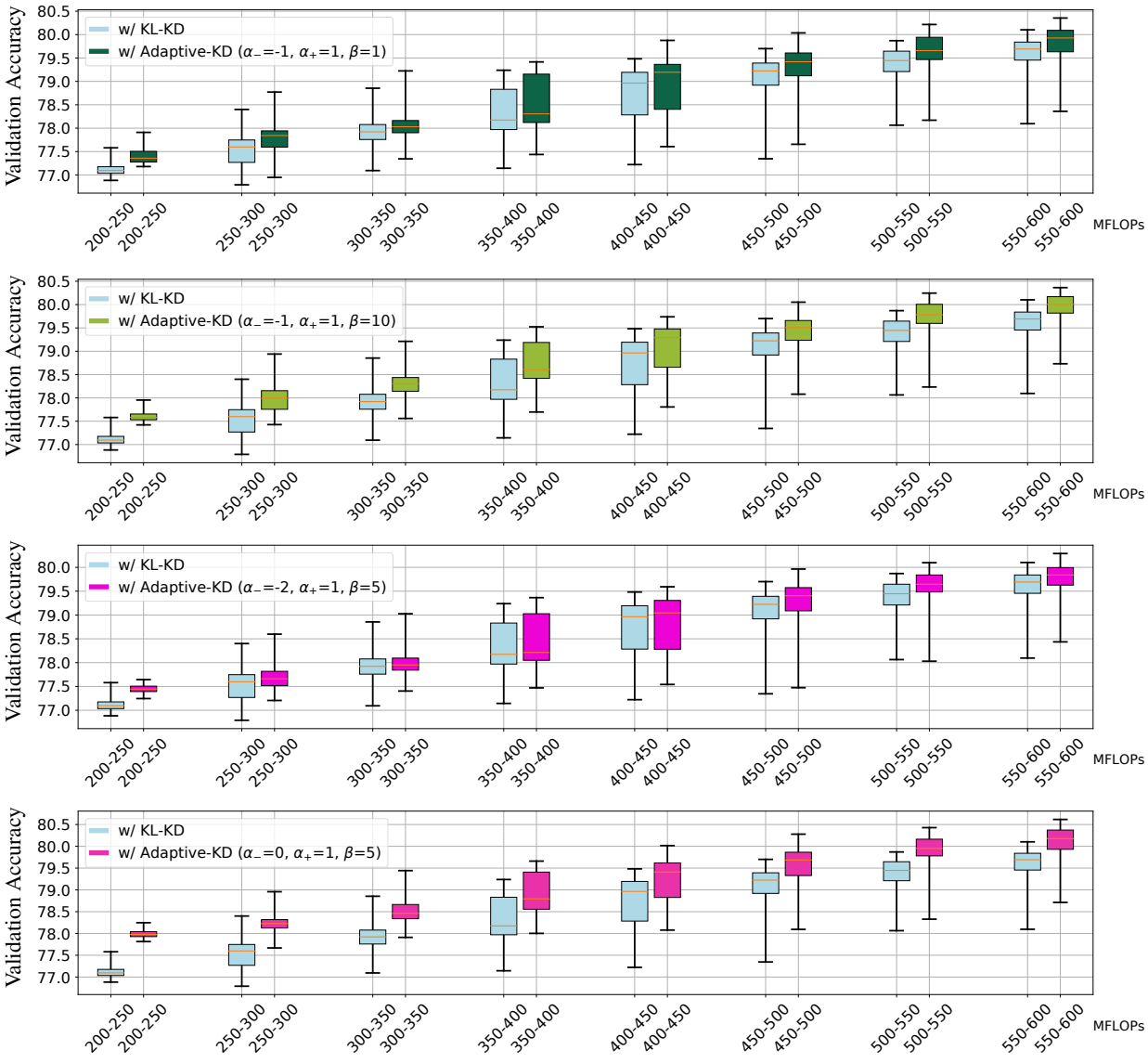


*Figure 9.* Additional results on ablation studies. Each box plot shows the performance of sampled sub-networks within each FLOPs regime. From bottom to top, each horizontal bar represents the minimum accuracy, the first quartile, the median, the third quartile and the maximum accuracy, respectively.

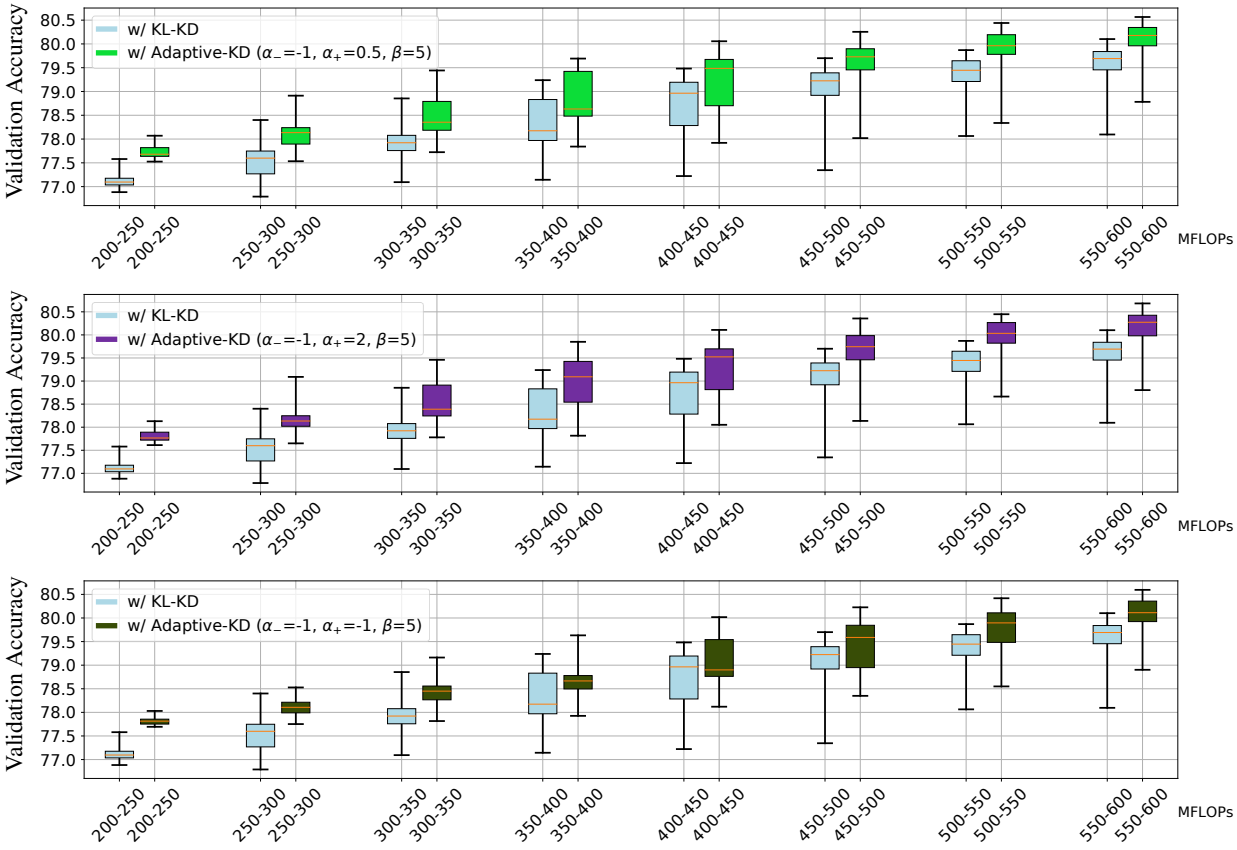


*Figure 10.* Additional results on ablation studies. Each box plot shows the performance of sampled sub-networks within each FLOPs regime. From bottom to top, each horizontal bar represents the minimum accuracy, the first quartile, the median, the third quartile and the maximum accuracy, respectively.