Landmark Regularization: Ranking Guided Super-Net Training in Neural Architecture Search
Supplementary Material

1. Implementation details

We report additional implementation details for each method.

1.1. Settings for landmark regularization

For the training, we randomly split the training set into two equally sized partitions, one to evaluate the super-net loss and another to evaluate the landmark regularization. Following [6], we disable the tracking of running statistics in the batch normalization layers. We sample $m = 1$ pairs for each super-net training step after the warm-up phase. For the CIFAR-10 experiments, we warm-up the super-net for 250 epochs before applying our landmark regularization.

ImageNet experiments. We use PyTorch's automatic mixed-precision mode to accelerate the training. All search experiments have a batch size of 256, while re-training having a size 1024 following [5]. For stand-alone training, we increase the learning rate from 0.1 to 0.25 in the first 5 epochs, then decrease to $1e-4$ in a linear fashion for 245 epochs.

1.2. Hyper-parameter Settings for three NAS algorithms

SPOS. After the super-net is trained, we randomly sample a population size of 50 architectures to begin the evolutionary search. We implement a budget of 1000 architectures to query the super-net accuracy. For the DARTS space, we retrain the top-1 architecture to evaluate the performance and repeat in total three times to ensure the robustness of our conclusions.

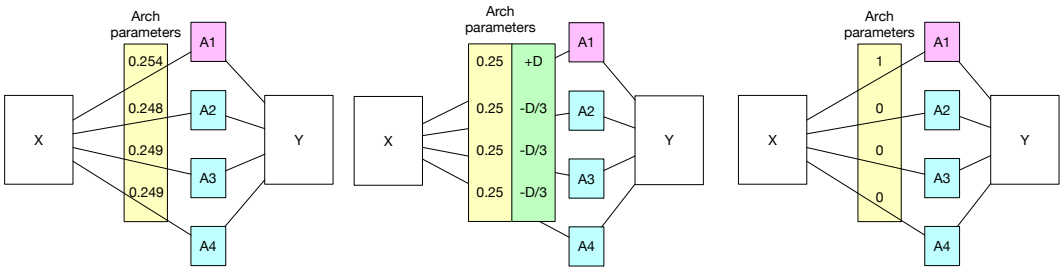
NAO. For both the encoder and the decoder, we set token embedding size to 48 for NASBench-101, to 12 for NASBench-201, and to 48 for the DARTS search space. We set the LSTM hidden layer size to 64 for all experiments. Each LSTM encoder has two layers. We train the controller for 1000 epochs at epochs 320, 360, and 400 in our CIFAR-10 experiments, and at epochs 140, 180, and 200 for our ImageNet experiments. Other hyper-parameters remain the same as in [4].

GDAS. We follow Dong and Yang [1]: We initialize the Gumbel-Softmax temperature τ to 10 and linearly decrease τ to 0.1 during training. We train the super-net in a SPOS fashion up to 400 epochs for CIFAR-10 experiments and up to 140 epochs for ImageNet experiments. We then train the architecture parameters for another 35 epochs. The initial learning rate is set to 0.1 for NASBench-201 and the DARTS search space on CIFAR-10 experiments, and 0.1 for ImageNet experiments. The learning rate follows a cosine annealing with a minimal value of $1e-5$. The batch size is set to 256 on NASBench-201 and 128 on DARTS. We set all other hyper-parameters, e.g. weight decay, learning rate scheduler, as proposed in [1].

1.3. Details of each search space

Introduce some special things for each search space. such as the number of architecture, etc. **NASBench-101.** This space restricts to search for the topology and operations to formulate a cell. Each cell is a fully connected direct acyclic graph, where the first node is the input and the last node represents the output node. In total, the number of nodes is set to 7. Each possible node has three operations, convolutional operation with kernel size 1 and 3, and max-pooling layer with kernel 3. It further limits the search space by pruning architectures with more than 9 active edges. We follow the adaptation in Yu et al. [7] to make NASBench-101 applicable to weight sharing NAS approaches. Regarding hyper-parameters, we set the channel number of the first cell to 64 and batch-size to 128.

NASBench-201. To address some limitations in NASBench-101 and better simulate the general NASNet like search space, Dong et al. [2] proposes NASBench-201. Similar to NASBench-101, it is a cell-based search space, whose cell encompasses four nodes including one input and one output one. Each edge of the graph consists of five operations: 1) zero, 2) skip-connection, 3) convolution with kernel size 1, 4) kernel size 3, and 5) average pooling with kernel size 3. For hyper-parameters, we set the batch size as 256 and the channel number at the first cell as 16.



(a) Continuous architecture selection (b) Soft One-Hot encoding (c) One-Hot Encoding

Figure 1: Different encodings of the continuous super-net. (a) Traditional continuous encoding used in DARTS-based method. The architecture is encoded in architecture parameters that sum to one. (b) We propose to select one architecture during continuous training by adding a fixed amount perturbation D to the selected branch, and subtracting a perturbation of $D/(n-1)$ from the other branches. (c) Naive approach to select one architecture by one-hot encoding. Note that, we refer (b) as soft one-hot encoding because it is in between of (a) and (c).

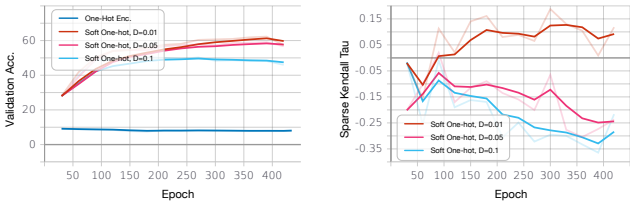


Figure 2: (Left) Validation accuracy during super-net training for one-hot encoding and our proposed soft one-hot with different D. We can observe that the naive one-hot encoding always yields accuracies around 0.1 on the validation set, whereas our proposed soft one-hot ranges from 50% to 60%. (Right) We further compare S-KdT. We choose $D = 0.01$ for further experiments with our landmark regularization.

DARTS search space. This is one of the most important NAS search spaces in the current literature [3, 5]. It originates from the micro search space in Zoph et al. [8]. It contains a DAG with 7 nodes, where two serving as previous input, one serving as an output node. The edge between the two input nodes is dropped. For each edge between the intermediate node to the input one consists of 8 operations to search, separable, and dilated convolutions with kernel size 3 and 5, max and average pooling with kernel size 3, identity, and zero. We search two types of cells, namely *normal* cell and *reduced* cell, which reduces the feature map dimension by a factor of 2. For CIFAR-10 experiments, we set the batch size to 64, the channel number of the first cell to 36, a total network depth of 12 searchable cells. For ImageNet experiments, we follow set batch-size to 256, the channel number of the first cell to 16, and a depth of 8 cells.

2. Additional result

We report additional results to further show the effectiveness of our method. We also show the discovered architectures for the image classification tasks.

Table 1: Additional results on NASBench-201. "Best" indicates the accuracy (with rank) of the best architecture out of three runs.

	CIFAR-100			ImageNet16-120		
Model	S-KdT	Mean Acc.	Best	S-KdT	Mean Acc.	Best
SPOS	0.506	66.81±0.32	69.51(1476)	0.637	40.28±3.61	44.68(418)
+Ours	0.558	67.49±0.28	70.92(238)	0.673	42.53±1.47	45.10(246)
GDAS	0.572	66.03±1.03	68.59(2836)	0.624	42.13±0.75	43.35(1235)
+Ours	0.677	67.74±0.63	69.67(1236)	0.709	43.50±0.61	44.83(351)
NAO	0.492	67.39±0.52	70.02(843)	0.615	40.44±1.35	41.86(2450)
+Ours	0.585	69.38±0.43	71.48(103)	0.631	43.02±1.01	44.27(613)

2.1. Additional Datasets of NASBench-201

We report comparisons on CIFAR-100 and ImageNet16-120 in Table 1. Note that all standard deviations of S-KdT are smaller than 0.03. Our approach also provides a consistent improvement on these datasets.

2.2. Case study: Application on PC-DARTS

We show how to apply our method to DARTS-based methods with a continuous sampling of the super-net. We pick the state-of-the-arts method PC-DARTS [5] to experimentally validate our approach. We conduct our experiments on NASBench-201 search space.

Naive one-hot encoding in continuous space. We introduce a novel soft one-hot encoding to apply our landmark regularization in a continuous super-net. A fundamental difference between discrete super-net with a continuous one is how to select a single architecture from the continuous architecture specification (also known as rounding), which is a requirement to be able to evaluate our regularization term. As shown in Figure 1(c), a single architecture in a discrete space can be picked directly from its associated weights. In continuous space, however, there is no real architecture selection because the architecture parameters are part of the

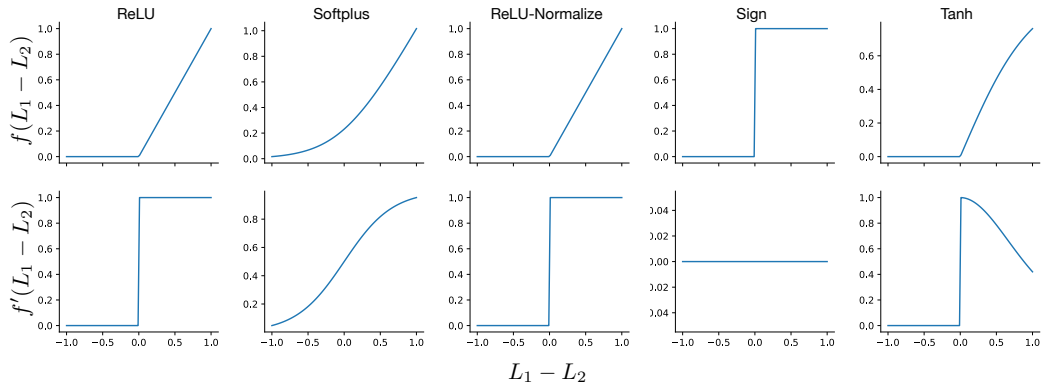


Figure 3: Visualization of different loss formulation $f(L_1 - L_2)$ and its first-order gradient. Note that the loss shape of ReLU and ReLU-Normalize are the same, because we normalize L_i but not its associated function.

Loss Name	ReLU	Softplus	ReLU-Norm	Sign	Tanh
S-KdT	0.802	0.613	0.741	0.709	0.756
Mean Acc.	92.08	92.00	91.91	91.02	91.48

Table 2: Validating different loss function on NASBench-201

super-net as shown in Figure 1(a). To the best of our knowledge, there is no existing method that can effectively sample one architecture from a continuous search space to conduct ranking analysis.

Soft one-hot encoding. To this end, we propose a simple but effective encoding to sample one architecture from a continuous super-net, dubbed as soft one-hot encoding. The idea is to relax the traditional one-hot encoding to simulate the continuous one. As shown in Figure 1(b), we start with a uniform distribution and add a perturbation D to the selected branch, while substracting a perturbation by $D/(n-1)$ from the other branches such that the sum of weights remains equal to one. In Figure 2 (left), we can see this drastically increases validation accuracy from 10% to around 50%. In addition, we ablate different values of $D \in \{0.01, 0.05, 0.1\}$ in Figure 2 (right). Setting $D = 0.01$ reaches the highest S-KdT.

Applying landmark regularization with the soft one-hot encoding. We further deploy this method with our landmark regularization. We keep all configuration the same as in our previous NASBench-201 GDAS experiment. Our regularization improves the S-KdTfrom 0.12 to 0.269, and improves the best model from rank 906 to 245. This evidences that our method can generalize to continuous differentiable architecture search method.

2.3. Loss formulation

We validate other possible formulations of the ranking regularization. Specifically, to penalize the architectures that disobey the ideal ranking order, we evaluate various instances of loss functions f that obey $\mathcal{R} = f(\mathcal{L}(x, \theta^s_{a_i}), \mathcal{L}(x, \theta^s_{a_j})) > 0$ when $i < j$ but $\mathcal{L}(x, \theta^s_{a_i}) > \mathcal{L}(x, \theta^s_{a_j})$. We shorten $\mathcal{L}(x, \theta^s_{a_i})$ to L_i during this discussion, and rewrite the formulation as $\mathcal{R} = f(L_i - L_j)$. We evaluate the following functions (*c.f.* Figure 3):

- *ReLU*: $f(x) = \max(0, x)$
- *Softplus*: $f(x) = \ln 1 + e^{kx}/k$, where $k = 3$
- *ReLU-normalized*: $f(x) = \max(0, x)$, however, we normalize L_i s.t. $L_i \in [0, 1]$.
- *Sign*: $f(x) = \max(0, \text{sign}(x))$
- *Tanh*: $f(x) = \max(0, \tanh(x))$

Table 2 reports the results of the different loss functions on NASBench-201. The best performance is achieved by the ReLU formalism, which corresponds to the formulation shown in the main paper.

2.4. Searched model

For DARTS search space, we report the configuration of the searched models on CIFAR-10 in Figure 4 and on ImageNet in Figure 5.

References

[1] Xuanyi Dong and Yi Yang. Searching for a robust neural architecture in four gpu hours. In *CVPR*, 2019. 1

[2] Xuanyi Dong and Yi Yang. NAS-Bench-201: Extending the scope of reproducible neural architecture search. In *ICLR*, 2020. 1

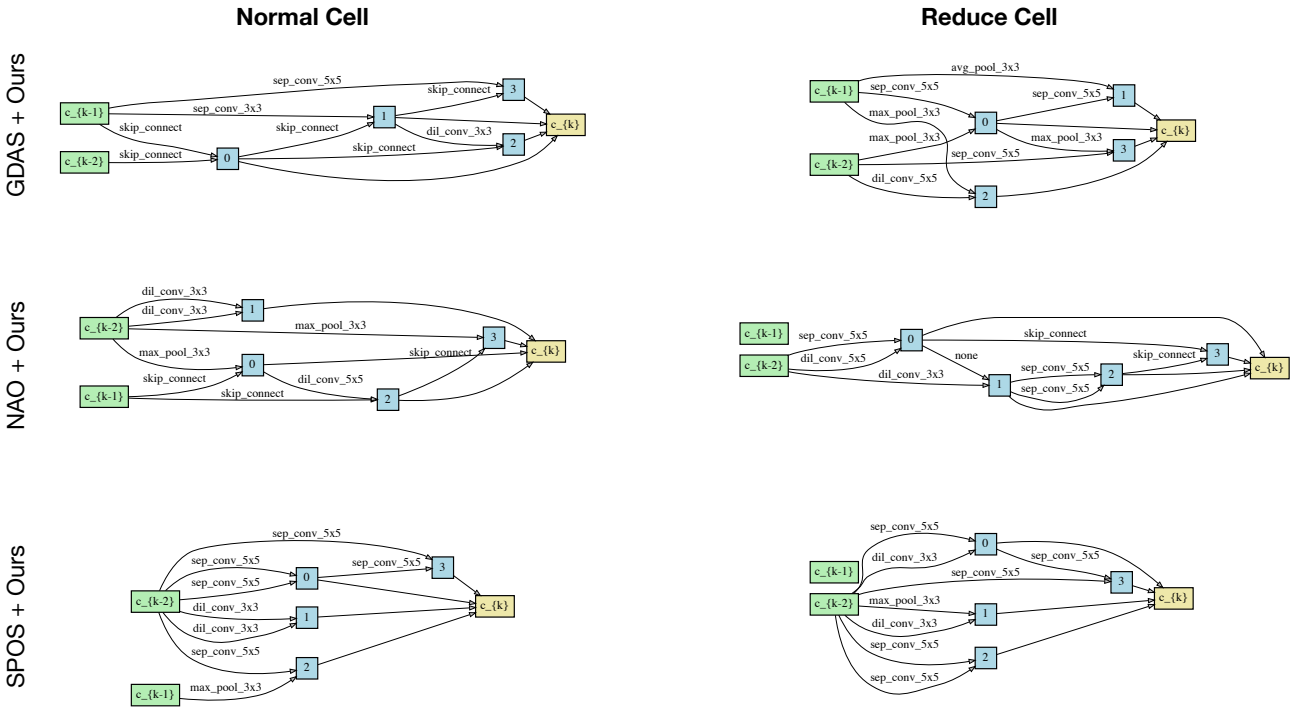


Figure 4: Best architectures discovered by our algorithms on CIFAR-10.

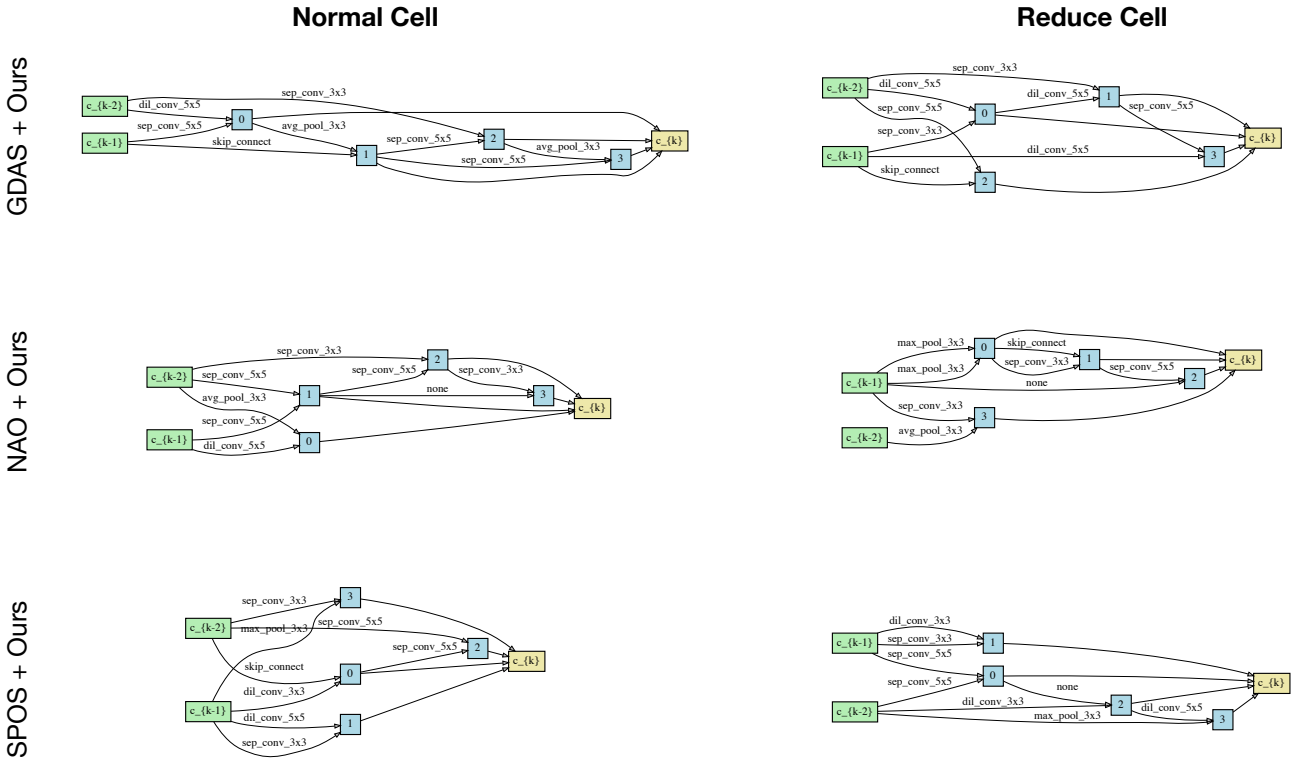


Figure 5: Best architectures discovered by our algorithms on ImageNet.

[3] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. *ICLR*, 2019. 2

[4] Renqian Luo, Fei Tian, Tao Qin, En-Hong Chen, and Tie-Yan Liu. Neural architecture optimization. In *NeurIPS*, 2018. 1

[5] Yuhui Xu, Lingxi Xie, Xiaopeng Zhang, Xin Chen, Guo-Jun Qi, Qi Tian, and Hongkai Xiong. PC-DARTS: Partial channel connections for memory-efficient architecture search. In *ICLR*, 2020. 1, 2

[6] Kaicheng Yu, Rene Ranftl, and Mathieu Salzmann. How to Train Your Super-Net: An Analysis of Training Heuristics in Weight-Sharing NAS. *arXiv*, 2020. 1

[7] Kaicheng Yu, Christian Sciuto, Martin Jaggi, Claudiu Musat, and Mathieu Salzmann. Evaluating the search phase of neural architecture search. In *ICLR*, 2020. 1

[8] Barret Zoph and Quoc V. Le. Neural Architecture Search with Reinforcement Learning. *ICLR*, 2017. 2