Neural Architecture Search in a Proxy Validation Loss Landscape

Yanxi Li[1], Minjing Dong[1], Yunhe Wang[2], Chang Xu[1]

[1]University of Sydney [2]Huawei Noah's Ark Lab.

Aim

Improve the efficiency of Neural Architecture Search (NAS) via learning a Proxy Validation Loss Landscape (PVLL) with historical validation results.

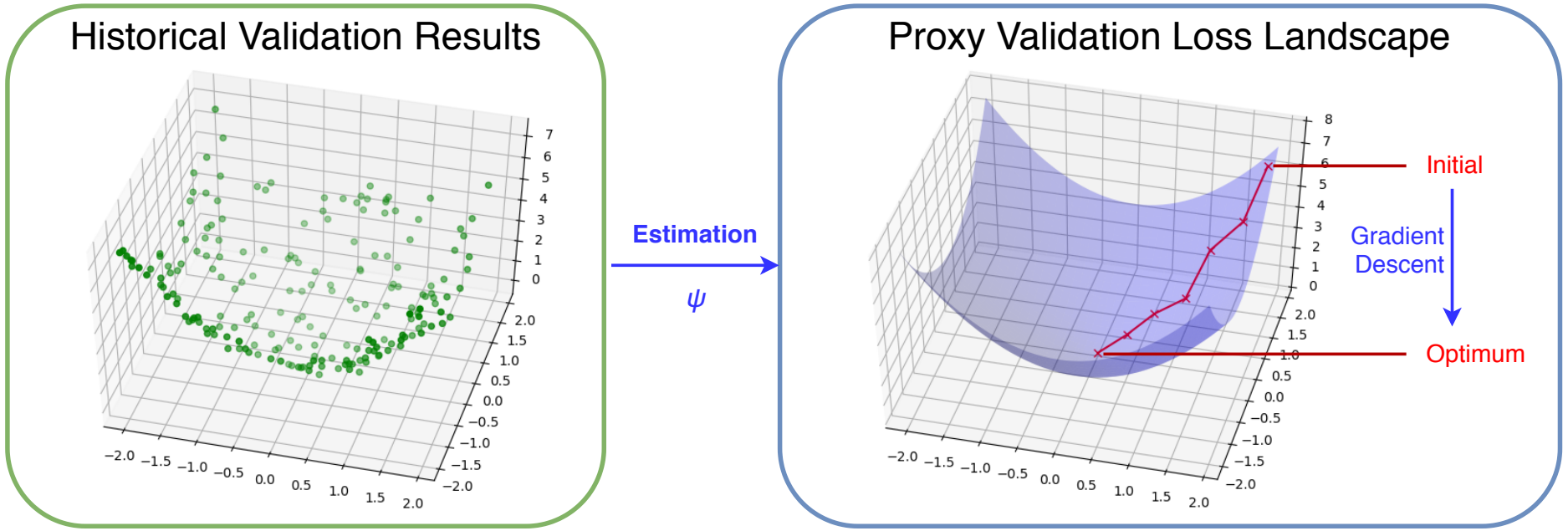
The Bi-level Setting of NAS

$$\begin{aligned} \min_{\boldsymbol{A}} \quad & \mathcal{L}(\mathbb{D}_{valid}; \boldsymbol{w}^*(\boldsymbol{A}), \boldsymbol{A}), \\ \mathbf{s.t.} \quad & \boldsymbol{w}^*(\boldsymbol{A}) = \arg\max_{\boldsymbol{w}} \mathcal{L}(\mathbb{D}_{train}; \boldsymbol{w}, \boldsymbol{A}). \end{aligned}$$

- The bi-level optimization is solved iteratively;
- When $\boldsymbol{\alpha}$ is updated, $\boldsymbol{w}^*(\boldsymbol{\alpha})$ also changes;
- $\boldsymbol{w}$ needs to be updated towards $\boldsymbol{w}^*(\boldsymbol{\alpha})$, and $\boldsymbol{\alpha}$ is evaluated again;
- In this process, intermediate validation results are used once and discarded.

Make Use of Historical Validation Results

Approach: learn a PVLL with them



PVLL-NAS

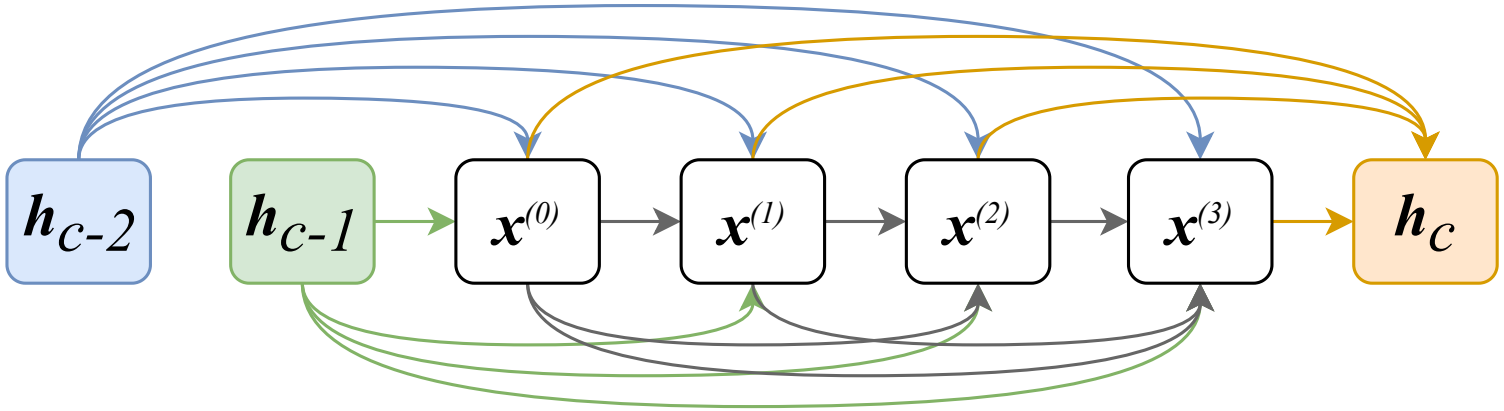
Advantages:

- Learning a Proxy Validation Loss Landscape (PVLL) with historical validation results;
- Sampling new architectures from the PVLL for further evaluation and update;
- Efficient architecture search with gradients of the PVLL.

Methodology

Search Space

A micro search space: the NASNet search space



$$\boldsymbol{I}^{(j)} = \sum_{i<j} o_{i,j}(\boldsymbol{I}^{(i)}), \quad \text{for } i = 2, 3, 4, 5.$$

$$o_{i,j} \in \mathcal{O}, \quad |\mathcal{O}| = K.$$

Operation Candidates

We use $K = 8$:

- 3×3 separable convolution;
- 5×5 separable convolution;
- 3×3 dilated separable convolution;
- 5×5 dilated separable convolution;
- 3×3 max pooling;
- 3×3 average pooling;
- Identity (i.e. skip-connection);
- Zero (i.e. not connected).

Select Operations

Calculate architecture parameters with Gumbel-Softmax:

$$\tilde{\boldsymbol{h}}_{i,j}^{(k)} = \frac{\exp\left((\boldsymbol{a}_{i,j}^{(k)} + \boldsymbol{\xi}_{i,j}^{(k)})/\tau\right)}{\sum_{k'=1}^{K} \exp\left((\boldsymbol{a}_{i,j}^{(k')} + \boldsymbol{\xi}_{i,j}^{(k')})/\tau\right)}.$$

Sample operations with argmax:

$$\boldsymbol{I}^{(j)} \approx \sum_{i<j} \tilde{\boldsymbol{h}}_{i,j}^{(k)} \cdot \mathcal{O}^{(k)}(\boldsymbol{I}^{(i)}),$$

$$\text{where} \quad k = \text{argmax}_k \, \tilde{\boldsymbol{h}}_{i,j}^{(k)}.$$

Evaluate Architectures

$$
\begin{aligned}
\min_{\boldsymbol{A}} \quad & \mathcal{L}(\mathbb{D}_{valid}; \boldsymbol{w}^*(\tilde{\boldsymbol{H}}), \tilde{\boldsymbol{H}}), \\
\textbf{s.t.} \quad & \boldsymbol{w}^*(\tilde{\boldsymbol{H}}) = \arg\max_{\boldsymbol{w}} \mathcal{L}(\mathbb{D}_{train}; \boldsymbol{w}, \tilde{\boldsymbol{H}}), \\
& \tilde{\boldsymbol{H}} = \mathrm{GumbelSoftmax}(\boldsymbol{A}; \boldsymbol{\xi}, \tau).
\end{aligned}
$$

Proxy Validation Loss Landscape

The PVLL is learned by learning a mapping ψ: $\widetilde{\boldsymbol{H}} \rightarrow \hat{\mathcal{L}}$;

$$\min_{\psi} \quad L_T(\psi) = \frac{1}{T} \sum_{t=1}^{T} \frac{1}{p_t} \left(\psi(\tilde{\boldsymbol{H}}_t) - \mathcal{L}_t \right)^2 .$$

Proxy Validation Loss Landscape

The PVLL is learned with a memory M, such that

$$\boldsymbol{M} = \{(\tilde{\boldsymbol{H}}_t, \boldsymbol{\mathcal{L}}_t), 1 \leq t \leq T\}.$$

After each sampling, the memory M is updated by:

$$\boldsymbol{M} = \boldsymbol{M} \cup \{(\tilde{\boldsymbol{H}}_t, \boldsymbol{\mathcal{L}}_t)\}.$$

Proxy Validation Loss Landscape

The next architecture is determined by the current architecture A and its gradients in the PVLL:

$$\boldsymbol{A}' \leftarrow \boldsymbol{A} - \eta \cdot \nabla_{\boldsymbol{A}} \psi_t^*(\tilde{\boldsymbol{H}}),$$

where A' is the next architecture and η is a learning rate.

Overall Algorithm

Algorithm 1 Loss Space Regression

Initialize a warm-up population:

$$\mathbf{P} = \{\tilde{\boldsymbol{H}}_i | i = 1, ..., N\}$$

for each $\tilde{\boldsymbol{H}}_i \in \mathbf{P}$ **do**
    Warm-up architecture $\tilde{\boldsymbol{H}}_i$ for 1 epoch
end for
Initialize a performance memory $\boldsymbol{M} = \emptyset$
for each $\tilde{\boldsymbol{H}}_i \in \mathbf{P}$ **do**
    Train architecture $\tilde{\boldsymbol{H}}_i$ for 1 epoch
    Evaluate architecture $\tilde{\boldsymbol{H}}_i$'s loss $\mathcal{L}_i$
    Set $\boldsymbol{M} = \boldsymbol{M} \cup \{(\tilde{\boldsymbol{H}}_i, \mathcal{L}_i)\}$
end for
Warm-up ψ with $\boldsymbol{M}$
for $t = 1 \rightarrow T$ **do**
    Sample an architecture as in Eq. 4 with $\tilde{\boldsymbol{H}}_t$:

$$\tilde{\boldsymbol{H}}_t = \text{GumbelSoftmax}(\boldsymbol{A}_t; \boldsymbol{\xi}_t, \tau)$$

    Optimize network with loss in Eq. 5
    Evaluate architecture to obtain loss $\mathcal{L}_t$
    Set $\boldsymbol{M} = \boldsymbol{M} \cup \{(\tilde{\boldsymbol{H}}_t, \mathcal{L}_t)\}$
    Update ψ with Eq. 8
    Update $\boldsymbol{A}_t$ to $\boldsymbol{A}_{t+1}$ with Eq. 10
end for

Theoretical Analysis

Theoretical Analysis

- The algorithm consistency;
- The label complexity.

Consistency of PVLL

Theorem 1. *Let Ψ be a hypothesis class containing all the possible hypothesises of estimator ψ. For any $\delta > 0$, with probability at lest $1 - \delta$, $\forall \psi \in \Psi$:*

$$|L_T(\psi) - L(\psi)| < \sqrt{\frac{2\left(d + \ln\frac{2}{\delta}\right)}{T}},$$

where d is the Pollard's pseudo-dimension of Ψ.

Label Complexity of PVLL

Theorem 2. *With probability at least* $1-\delta$*, to learn an estimator* ψ *with error bound* $\epsilon \leq \sqrt{(8/N)(d+\ln(2/\delta))}$*, the number of labels requested by the algorithm is at most the order of*

$$\mathcal{O}\left(\sqrt{N(d+\ln\left(2/\delta\right))}\right).$$

Experiments

Search and Evaluate on CIFAR-10

We search for architectures on CIFAR-10. Firstly, 100 random architectures are sampled for the warm-up of PVLL. Then, we search for 100 steps in the PVLL.

Model	GPUs	Time (Days)	Params (M)	Test Error (%)
ResNet-110	-	-	1.7	6.61
DenseNet-BC	-	-	25.6	3.46
MetaQNN	10	8-10	11.2	6.92
NAS	800	21-28	7.1	4.47
NAS+more filters	800	21-28	37.4	3.65
ENAS	1	0.32	21.3	4.23
ENAS+more channels	1	0.32	38.0	3.87
NASNet-A	450	3-4	3.3	3.41
NASNet-A+cutout	450	3-4	3.3	2.65
ENAS	1	0.45	4.6	3.54
ENAS+cutout	1	0.45	4.6	2.89
DARTS(1st)+cutout	1	1.50	3.3	3.00
DARTS(2nd)+cutout	1	4	3.3	2.76
NAONet+cutout	200	1	128	2.11
NAONet+WS	1	0.30	2.5	3.53
GDAS	1	0.21	3.4	3.87
GDAS+cutout	1	0.21	3.4	2.93
PVLL-NAS	1	0.20	3.3	2.70

Table 1. Comparison of PVLL-NAS with different state-of-the-art CNN models on CIFAR-10 dataset.

Generalize to ImageNet

Architectures found on CIFAR-10 is generalized to ImageNet for evaluation. Evaluation on ImageNet follows the mobile setting, i.e. no more than 600 multi-add operations.

Model	GPUs	Time (Days)	Params (M)	+× (M)	Test Error (%)	
					Top-1	Top-5
Inception-V1	-	-	6.6	1448	30.2	10.1
MobileNet-V2	-	-	3.4	300	28.0	-
ShuffleNet	-	-	~ 5	524	26.3	-
Progressive NAS	100	1.5	5.1	588	25.8	8.1
NASNet-A	450	3-4	5.3	564	26.0	8.4
NASNet-B	450	3-4	5.3	488	27.2	8.7
NASNet-C	450	3-4	4.9	558	27.5	9.0
AmoebaNet-A	450	7	5.1	555	25.5	8.0
AmoebaNet-B	450	7	5.3	555	26.0	8.5
AmoebaNet-C	450	7	6.4	570	24.3	7.6
DARTS	1	4	4.9	595	26.7	8.7
GDAS	1	0.21	5.3	581	26.0	8.5
PVLL-NAS	1	0.20	4.8	532	25.6	8.1

Table 2. Top-1 and top-5 error rates of PVLL-NAS and other state-of-the-art cnn models on ImageNet dataset.

Ablation Test - Estimation Strategies

Some differentiable NAS methods use the 2nd order estimation for better gradients. We demonstrate that the gradients estimated by PVLL is also competitive.

Method	Order	Time (Days)	Test Error (%)
DARTS	1st	1.5	3.00 ± 0.14
	2nd	4.0	2.76 ± 0.09
Amended-DARTS	1st	-	-
	2nd	1.0	2.81 ± 0.21
PVLL-NAS	1st	0.10	3.48
	2nd	0.20	2.72 ± 0.02

Table 3. Performances of architectures found on CIFAR-10 with different order of approximation.

Ablation Test - Sampling Strategies

Different sampling strategies are tested, including using warm-up or not, using weighted loss or not, and using a uniform sampler.

With Sampler	Warm-up	Weighted Loss	Test Error (%)
Y	Y	Y	2.72 ± 0.02
Y	Y	N	2.81 ± 0.08
Y	N	Y	3.10 ± 0.22
Y	N	N	3.03 ± 0.30
N	Y	N/A	3.08 ± 0.24
N	N	N/A	3.20 ± 0.32

Table 4. Ablation studies on the performances of architectures searched on CIFAR-10 with different strategies.

Conclusion

Conclusion

In this paper, we propose to search for neural architectures with a proxy validation loss landscape. We introduce a novel method to dynamically sample architecture to be evaluated for the efficient validation loss estimator training. Both theoretical analysis and experiments show that this approach can establish a satisfactory proxy validation loss landscape with less computational resource. Experimental results demonstrate that the proposed NAS algorithm can efficiently design networks of the competitive performance compared to state-of-the-art methods.

Thank You!