Learning Where To Look - Generative NAS is surprisingly efficient

Jovita Lukasik*, Steffen Jung*, Margret Keuper

max planck institut informatik

UNIVERSITY OF MANNHEIM
School of Business Informatics and Mathematics

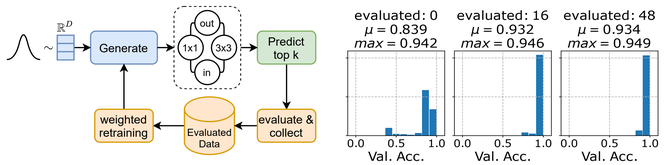
UNIVERSITÄT SIEGEN

ECCV TEL AVIV 2022

Motivation / Contribution

Existing neural architecture search (NAS) methods are expensive and require to evaluate numerous architectures for their performance. So, how can we find well-performing architectures efficiently?

Our idea: We finetune the latent space of a generative model in a weighted retraining manner such that it learns to directly generate well-performing architectures



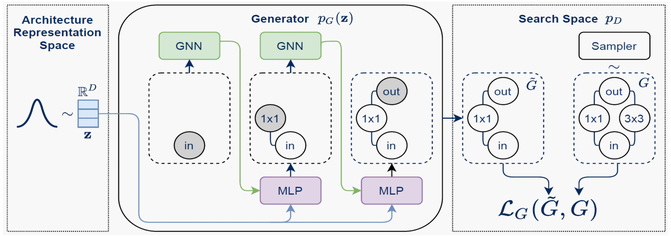
Our contributions are as follows:

1. Efficient generative model without an encoder network.
2. First algorithm optimizing architecture latent space in NAS directly.
3. State-of-the-art results on several NAS benchmarks and on ImageNet.

1) Generator Training

Generator $p_{G(z)}$ takes a randomly sampled variable $z \sim N(0,1)$ as input and learns to reconstruct a randomly sampled graph G from the architecture search space p_D.

Training procedure: The generator generates in the first step the input node, which is then the input to a GNN. This GNN generates a partial graph representation. The input variable z and the latter graph representation are then used for an iterative generation process to generate node and edge scores until the output node is generated. The output graph is then compared to the randomly sampled graph.



2) Generative Search Algorithm - AG-Net

We use the pretrained generator to generate architectures and use a surrogate prediction model to predict architecture performance or latency.

1. The surrogate model is trained together with the generator on a small initial training data set (16 training instances) for the combined objective of reconstruction and prediction in a weighted retraining manner.
2. The generator generates a set of unique architectures.
3. The surrogate model predicts e.g. the architecture's performance.
4. The predicted best top k architectures are selected for ground truth evaluation and added to the training data set.

This iterative training and evaluation process stops until a maximal query budget (300 architectures) is reached.

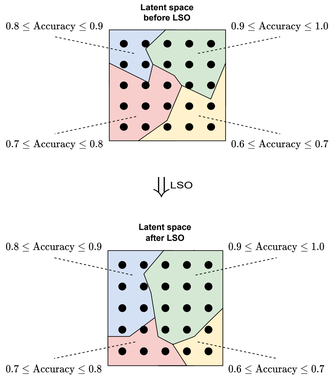
Algorithm 1: Unconstrained Search Algorithm

Input: (i) Search space p_D
Input: (ii) Pretrained generator G
Input: (iii) Untrained performance predictor P
Input: (iv) Query budget b
Input: (v) e epochs to train G and P
▷ Initialize training data
$\mathbf{D} \leftarrow \{\}$
while $|\mathbf{D}| < 16$ **do**
  $\mathbf{D} \leftarrow \mathbf{D} \cup \{d \sim p_D\}$
end
▷ Evaluate architectures (get accurracies on target image dataset)
$\mathbf{D} \leftarrow \text{eval}(\mathbf{D})$
▷ Randomly initialize predictor weights
$P \leftarrow \text{init}(P)$
▷ Search loop
while $|\mathbf{D}| < b$ **do**
  ▷ Weight training data by| performance
  $\mathbf{D}_w \leftarrow \text{weight}(\mathbf{D})$
  ▷ Train generator and predictor
  $\text{train}(G, P, \mathbf{D}_w, e)$
  ▷ Generate 100 candidates
  $\mathbf{D}_{\text{cand}} \leftarrow \{\}$
  while $|\mathbf{D}_{\text{cand}}| < 100$ **do**
    $\mathbf{z} \sim \mathcal{U}[-3, 3]$
    $\mathbf{D}_{\text{cand}} \leftarrow \mathbf{D}_{\text{cand}} \cup G(\mathbf{z})$
  end
  ▷ Select top 16 candidates with P
  $\mathbf{D}_{\text{cand}} \leftarrow \text{select}(\mathbf{D}_{\text{cand}}, P, 16)$
  ▷ Evaluate and add to data
  $\mathbf{D} \leftarrow \mathbf{D} \cup \text{eval}(\mathbf{D}_{\text{cand}})$
end

3) Latent Space Optimization

Weighted retraining [1] is used for latent space optimization. By weighting each training data by a weight w we place more probability mass on points (in this case architectures with high accuracy). Important to note here ist, that LSO does not discard low-scoring points from the training set, which would be inadequate for proper learning.

$$w(G; p_D, k) \propto \frac{1}{kN + \text{rank}_{f,p_D}(G)}$$
$$\text{rank}_{f,p_D}(x) = |\{G_i : f(G_i) > f(G), G_i \sim p_D\}|$$

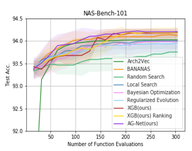


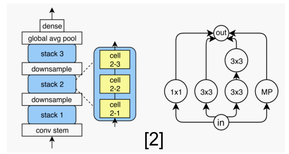
Paper

Code

4) Experiments - Single Objective

1. **NAS-Bench-101**: 423,624 architectures evaluated on CIFAR-10



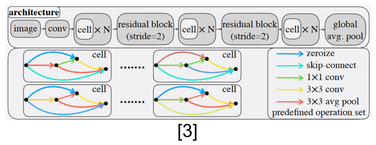


[2]

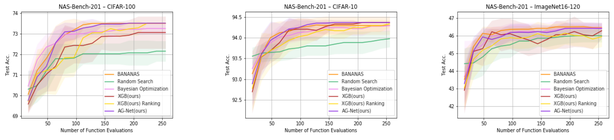
AG-Net improves over all SoTA methods not only at the last query of 300 data points, but is also almost any time better during the search process.

2. **NAS-Bench-201**: 15,625 architectures evaluated on CIFAR-10, CIFAR-100, ImageNet16

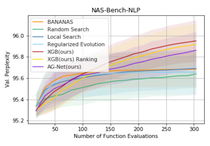
AG-Net provides SoTA results on all three datasets.

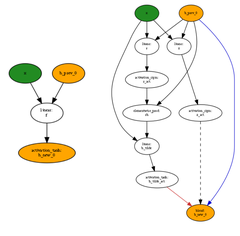


[3]



3. **NAS-Bench-NLP:** 14,322 sampled and evualated architectures on PTB in an RNN-derived benchmark (10^{53})





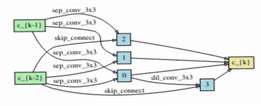
[4]

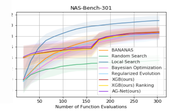
Evaluation on validation perplexity. AG-Net improves over all sota methods by a substantial margin. Note, surrogate benchmark used for the search process [5].

4) Experiments - Single Objective

4. **NAS-Bench-301 / DARTS**:~60k sampled and evaluated architectures on CIFAR-10 in the DARTS search space (10^{18})

Search in the DARTS search space based on the surrogate Benchmark NAS-Bench-301 [7]. Our method is comparable to sota methods





[6]

ImageNet experiments: Instead of searching and evaluating 300 different architectures use a different approach.

1. retrain best found architectures from NAS-Bench-301.
2.Use training-free measurements from TENAS [8] (neural tangent kernel (NTK) and number of linear regions (NLR)).

As a result our method is highly efficient and outperforms other methods in terms of needed GPU days. Using the training-free methods even improves over top-1 and top-5 error rates.

Table 4: ImageNet **error** of neural architecture search on DARTS.

NAS Method	Top-1↓	Top-5↓	# Queries	Search GPU days
Mixed Methods				
NASNET-A (CIFAR-10) [75]	26.0	8.4	20000	2000
PNAS (CIFAR-10) [33]	25.8	8.1	1160	225
NAO (CIFAR-10) [36]	24.5	7.8	1000	200
Differentiable Methods				
DARTS (CIFAR-10) [34]	26.7	8.7	-	4.0
SNAS (CIFAR-10) [61]	27.3	9.2	-	1.5
PDARTS (CIFAR-10) [12]	**24.4**	**7.4**	-	0.3
PC-DARTS (CIFAR-10) [63]	25.1	7.8	-	**0.1**
PC-DARTS (ImageNet) [63]	**24.2**	**7.3**	-	3.8
Predictor Based Methods				
WeakNAS (ImageNet) [60]	**23.5**	**6.8**	800	2.5
XGB (NB-301)(CIFAR-10) (ours)	24.1	7.4	304	**0.02**
XGB + Ranking (NB-301)(CIFAR-10) (ours)	24.1	7.2	304	**0.02**
AG-Net (NB-301)(CIFAR-10) (ours)	24.3	7.3	304	0.21
Training-Free Methods				
TE-NAS (CIFAR-10) [11]	26.2	8.3	-	0.05
TE-NAS (ImageNet) [11]	24.5	7.5	-	0.17
AG-Net (CIFAR-10) (ours)	**23.5**	7.1	208	**0.02**
AG-Net (ImageNet) (ours)	**23.5**	6.9	208	**0.09**

5) Experiments - Multi Objective

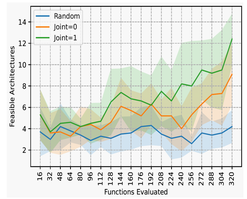
Hardware-Aware NAS-Bench: NAS-Bench-201 with latency information for different devices [9]

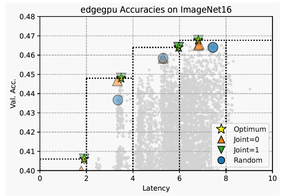
Joint optimization (joint=1): high accuracy and low latency.

$$\max_{G \sim p_D} f(G) \wedge \min_{G \sim p_D,} g_h(G)$$
$$\text{s.t. } g_h(G) \leq L, \exists\, h \in H,$$

Constrained optimization (joint=0): high accuracy and constrained latency.

$$\max_{G \sim p_D} f(G)$$
$$\text{s.t. } g_h(G) \leq L, \exists\, h \in H,$$





Conclusion

Strengths

- Adaptable to different search spaces and objectives
- Fast: couple of minutes on CPU for the search
- Sample-efficient

Weaknesses

- Pretraining of generator for each search space
- Sensitive to predictor performance

References

[1] A. Tripp, E. Daxberger, and J. Hernandez-Lobato. Sample efficient optimization in the latent space of deep generative models via weighted retraining. In NeurIPS, 2020.

[2] C. Ying, A. Klein, E. Christiansen, E. Real, K. Murphy, and F. Hutter. Nas-bench-101: Towards reproducible neural architecture search. In ICML, 2019.

[3] X. Dong, Y. Yang. NAS-Bench-201: Extending the Scope of Reproducible Neural Architecture Search. In ICLR, 2020.

[4] N. Klyuchnikov, I. Trofimov, E. Artemova, M. Salnikov, M. V. Fedorov, E. Burnaev: NAS-Bench-NLP: Neural Architecture Search Benchmark for Natural Language Processing. In IEEE Access, 2022

[5] S. Yan, C. White, Y. Savani, F. Hutter. Nas-bench-x11 and the power of learning curves. in NeurIPS, 2021

[6] H. Liu, K. Simonyan, Y. Yang. DARTS: Differentiable Architecture Search. In ICLR, 2019

[7] A. Zela, J. N. Siems, L. Zimmer, J. Lukasik, M. Keuper, F. Hutter. Surrogate NAS Benchmarks: Going Beyond the Limited Search Spaces of Tabular NAS Benchmarks. In ICLR, 2022

[8] W. Chen, X. Gong, Z. Wang. Neural architecture search on imagenet in four GPU hours: A theoretically inspired perspective. In ICLR, 2021)

[9] C. Li, Z. Yu, Y. Fu, Y. Zhang, Y. Zhao, H. You, Q. Yu, Y. Wang, C. Hao, Y. Lin. Hw-nas-bench: Hardware-aware neural architecture search benchmark. In ICLR, 2021