BayesNAS: A Bayesian Approach for Neural Architecture Search

Hongpeng Zhou[1], Minghao Yang[1], Jun Wang[2], Wei Pan[1]

1. Department of Cognitive Robotics, Delft University of Technology, Netherlands
2. Department of Computer Science, University College London, UK
Correspondence to: Wei Pan <wei.pan@tudelft.nl>

Outline

- What we achieve
- Why we study
- How to realize
- Experiment
- Conclusion and future work

Outline

- What we achieve
- Why we study
- How to realize
- Experiment
- Conclusion and future work

What?

What are the highlights of this paper?

- Fast:
  Find the architecture on CIFAR-10 within *only 0.2 GPU days* using a *single GPU.*

- Simple:
  Train the overparameterized network for only *one epoch* then update the architecture.

- First Bayesian method for one-shot NAS:
  Apply Laplace approximation;
  Propose *fast Hessian calculation methods* for convolutional layers.

- Dependencies between nodes:
  Model dependencies between nodes *ensuring a connected derived graph*.

Outline

- What we achieve
- Why we study
- How to realize
- Experiment
- Conclusion and future work

Why?

- ## Why use one shot method?
  - Reduce search time without separate training, compared with reinforcement learning, neuroevolutionary approach;
  - NAS is treated as Network Compression.

- ## Why employ Bayesian learning?
  - It could prevent overfitting and does not require tuning a lot of hyperparameters;
  - Hierarchical sparse priors can be used to model the architecture parameters;
  - The priors can promote sparsity and model the dependency between nodes.

- ## Why apply Laplace approximation?
  - Easy implementation;
  - Close relationship between Hessian metric and network compression;
  - Acceleration effect to training convergence by second order optimization algorithm.

[1] MacKay, D. J. A practical bayesian framework for backpropagation networks. Neural computation, 4(3):448–472, 1992b.
[2] LeCun, Y., Denker, J. S., and Solla, S. A. Optimal brain damage. In Advances in neural information processing systems, pp. 598–605, 1990.
[3] Botev, A., Ritter, H., and Barber, D. Practical gauss-newton optimisation for deep learning. ICML, 2017.

Why?

- ## Why consider dependency?

- Most current one-shot methods disregard the dependencies between a node and its predecessors and successors, which may results in a disconnected graph.

- ## Example:

If node 2 is redundant, the expected graph has no connection from node 2 to 3 and from node 2 to 4.

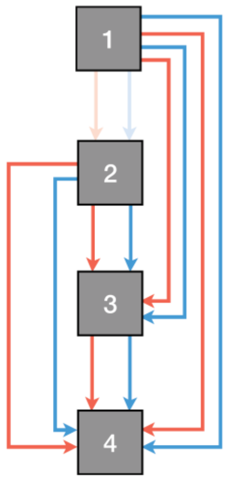


Figure1. Disconnected graph caused by disregard for dependency

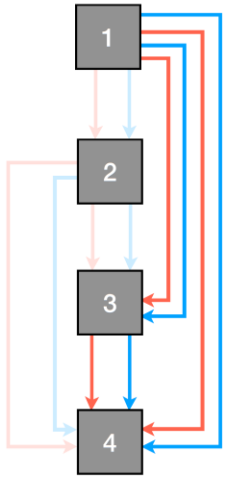


Figure2: Expected connected graph

Outline

- What we achieve
- Why we study
- How to realize
- Experiment
- Conclusion and future work

How?

- How to realize dependency?

A **multi-input-multi-output motif** is abstract the building block of any Directed Acyclic Graph (DAG). Any path or network can be constructed by this motif, as shown in Figure4.(c).

Proposition for Dependency: there is information flow from node j to k if and only if at least one operation of at least one predecessor of node j is non-zero and w_{jk}^{o} is also nonzero.

Specific explanation:

- *Figure3(a):* predecessor's (e_{12}) has superior control over its successors (e_{23} and e_{24});
- *Figure3(b):* design switches s_{12}, s_{23} and s_{24} to determine "on or off" of the edge;
- *Figure3(d):* prioritize zero operation over other non-zero operations by adding one more node i' between node i and j.

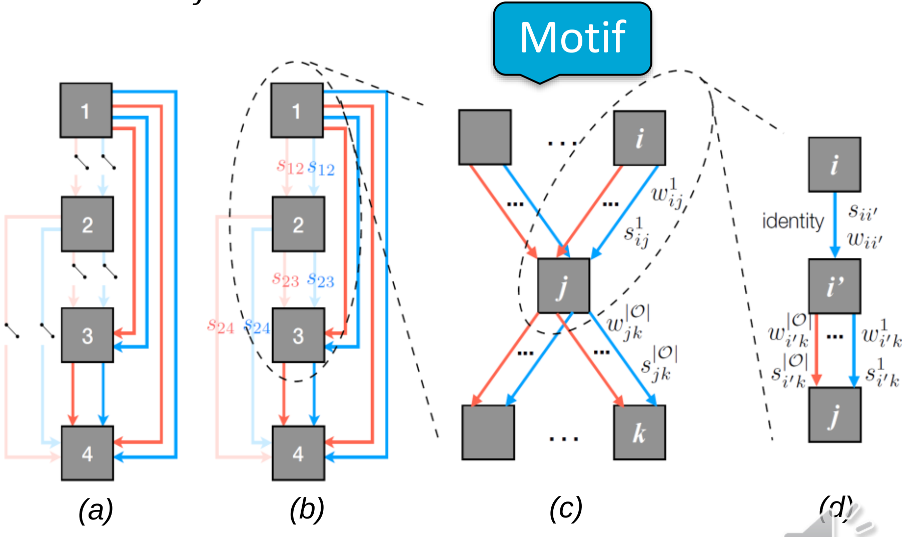


Figure3. An illustration for dependency.

How?

- ## How to apply Bayesian learning search strategy?

- Model architecture parameters with hierarchical automatic relevance determination (HARD) priors.

$$p(\mathbf{w} \,|\, \mathbf{s}) = \prod_{j<k} \prod_{o \in \mathcal{O}} \prod_{o' \in \mathcal{O}} \mathcal{N}\left(w_{jk}^{o'} \sum_{i<j} w_{ij}^{o} | 0, \gamma_{jk}^{o'} \right)$$

- The cost function is maximum likelihood over the data D with regularization whose intensity is controlled by the reweighted coefficient ω:

$$\mathcal{L}_D = E_D(\cdot) + \lambda_w \sum_{j<k} \sum_{o' \in \mathcal{O}} \|\omega_{jk}^{o'}(t) w_{jk}^{o'}\|_1 + \lambda \|\mathcal{W}\|_2^2$$

Loss on data

Regularization on architecture parameter

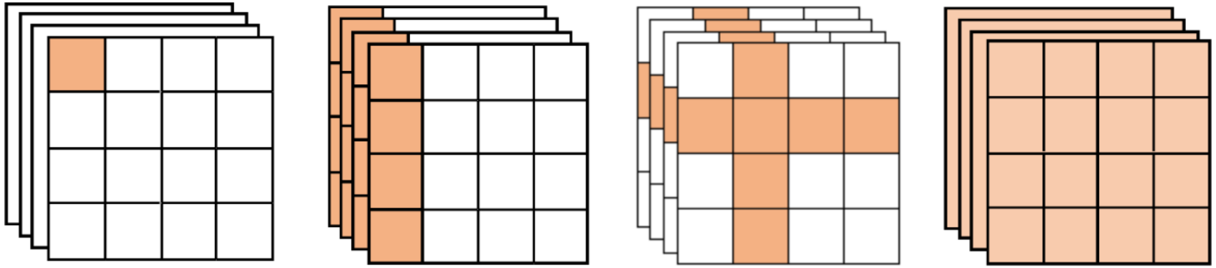
Regularization on Network parameter

- ## How to compute the Hessian?

- By converting convolutional layers to fully-connected layers, a recursive and efficient method is proposed to compute the Hessian of convolutional layers and architecture parameter.

Byproduct:

• Extension to Network Compression



$$\prod_{c_l}\prod_{m_l}\prod_{k_l}\mathcal{N}(\mathbf{0},\gamma_{c_l,m_l,k_l}\mathbf{I}_{n_l}) \qquad \prod_{k_l}\mathcal{N}(\mathbf{0},\gamma_{k_l}\mathbf{I}_{n_l c_l m_l}) \qquad \prod_{m_l k_l}\mathcal{N}(\mathbf{0},\gamma_{m_l k_l}\mathbf{I}_{n_l c_l}) \qquad \prod_{n_l}\mathcal{N}(\mathbf{0},\gamma_{n_l}\mathbf{I}_{c_l m_l k_l})$$

Figure4. Structure sparsity

- By enforcing various structural sparsity, extremely sparse models can be obtained without accuracy loss.
- This can be effortlessly integrated into BayesNAS to find sparse architecture for resource-limited hardware.

Outline

- What we achieve
- Why we study
- How to realize
- Experiment
- Conclusion and future work

Experiment:

• CIFAR10-experiment setting:

- The setup for proxy tasks follows DARTS and SNAS;
- The backbone for proxyless search is PyramidNet;
- Apply BayesNAS to search the best convolutional cells/optimal paths in a complete network;
- A network constructed by stacking learned cells/paths is retrained.

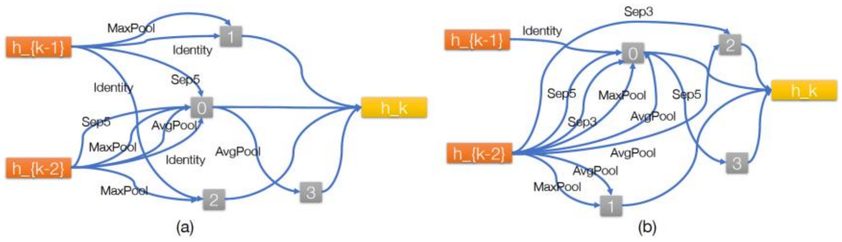


Figure 5. Normal and reduction cell found in proxy task

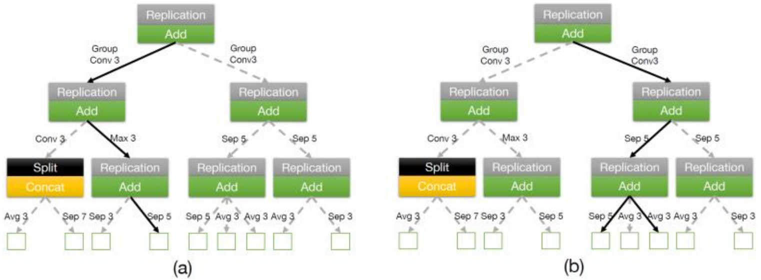


Figure 6. Tree cells found in proxyless task

[4] Liu, H., Simonyan, K., and Yang, Y. DARTS: Differentiable architecture search. ICLR, 2019b.
[5] Xie, S., Zheng, H., Liu, C., and Lin, L. SNAS: stochastic neural architecture search. ICLR, 2019.
[6] Cai, H., Zhu, L., and Han, S. ProxylessNAS: Direct neural architecture search on target task and hardware. ICLR, 2019.
[7] Cai, H., Yang, J., Zhang, W., Han, S., and Yu, Y. Path-level network transformation for efficient architecture search. ICML, 2018.

Experiment:

- ## CIFAR10-result:

Table 1. Classification errors of BayesNAS and state-of-the-art image classifiers on CIFAR-10.

Architecture	Test Error (%)	Params (M)	Search Cost (GPU days)	Search Method
DenseNet-BC (Huang et al., 2017)	3.46	25.6	-	manual
NASNet-A + cutout (Zoph et al., 2018)	2.65	3.3	1800	RL
AmoebaNet-B + cutout (Real et al., 2019)	2.55 ± 0.05	2.8	3150	evolution
Hierarchical Evo (Liu et al., 2018b)	3.75 ± 0.12	15.7	300	evolution
PNAS (Liu et al., 2018a)	3.41 ± 0.09	3.2	225	SMBO
ENAS + cutout (Pham et al., 2018)	2.89	4.6	0.5	RL
Random search baseline + cutout (Liu et al., 2019b)	3.29 ± 0.15	3.2	1	random
DARTS (2nd order bi-level) + cutout (Liu et al., 2019b)	2.76 ± 0.09	3.4	1	gradient
SNAS (single-level) + moderate con + cutout (Xie et al., 2019)	2.85 ± 0.02	2.8	1.5	gradient
DSO-NAS-share+cutout (Zhang et al., 2019b)	2.84 ± 0.07	3.0	1	gradient
Proxyless-G + cutout (Cai et al., 2019)	2.08	5.7	-	gradient
BayesNAS + cutout + $\lambda_w^o = 0.01$	3.02 ± 0.04	2.59 ± 0.23	0.2	gradient
BayesNAS + cutout + $\lambda_w^o = 0.007$	2.90 ± 0.05	3.10 ± 0.15	0.2	gradient
BayesNAS + cutout + $\lambda_w^o = 0.005$	2.81 ± 0.04	3.40 ± 0.62	0.2	gradient
BayesNAS + TreeCell-A + Pyrimaid backbone + cutout	2.41	3.4	0.1	gradient

less search time

- Competitive test error rate against state-of-the-art techniques.
- Significant drop in search time.

Experiment:

- ## Transferability to ImageNet :

A network of 14 cells is trained for 250 epochs with batch size 128:

Table 2. Comparison with state-of-the-art image classifiers on ImageNet in the mobile setting.

Architecture	**Test Error (%)**		**Params (M)**	**Search Cost (GPU days)**	**Search Method**
	top-1	top-5			
Inception-v1 (Szegedy et al., 2015)	30.2	10.1	6.6	–	manual
MobileNet (Howard et al., 2017)	29.4	10.5	4.2	–	manual
ShuffleNet 2× (v1) (Zhang et al., 2018)	29.1	10.2	∼5	–	manual
ShuffleNet 2× (v2) (Zhang et al., 2018)	26.3	–	∼5	–	manual
NASNet-A (Zoph et al., 2018)	26.0	8.4	5.3	1800	RL
NASNet-B (Zoph et al., 2018)	27.2	8.7	5.3	1800	RL
NASNet-C (Zoph et al., 2018)	27.5	9.0	4.9	1800	RL
AmoebaNet-A (Real et al., 2019)	25.5	8.0	5.1	3150	evolution
AmoebaNet-B (Real et al., 2019)	26.0	8.5	5.3	3150	evolution
AmoebaNet-C (Real et al., 2019)	24.3	7.6	6.4	3150	evolution
PNAS (Liu et al., 2018a)	25.8	8.1	5.1	∼225	SMBO
DARTS (Liu et al., 2019b)	26.9	9.0	4.9	4	gradient
BayesNAS ($\lambda_w^o = 0.01$)	28.1	9.4	4.0	0.2	gradient
BayesNAS ($\lambda_w^o = 0.007$)	27.3	8.4	3.3	0.2	gradient
BayesNAS ($\lambda_w^o = 0.005$)	26.5	8.9	3.9	0.2	gradient

Outline

- What we achieve
- Why we study
- How to realize
- Experiment
- Conclusion and future work

Conclusion and future work:

- First Bayesian approach for one-shot NAS: BayesNAS can prevent overfitting, promote sparsity and model dependencies between nodes ensuring a connected derived graph.

- Simple and fast search: BayesNAS is an iteratively re-weighted l1 type algorithm. Fast Hessian calculation methods are proposed to accelerate the computation. Only one epoch is required to update hyper-parameters.

- Our current implementation is still inefficient by caching all the feature maps in memory. The searching time could be future reduced by computing Hessian with backpropagation.

Thank you!

Paper: 3866

Contact: Wei Pan <wei.pan@tudelft.nl>