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 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.

• Why consider dependency?

• Most current one-shot methods disregard the dependencies between a node and its predecessors and successors, which may result 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.

*Figure 1.* Disconnected graph caused by disregard for dependency

*Figure 2:* Expected connected graph
Outline

• What we achieve
• Why we study
• How to realize
• Experiment
• Conclusion and future work
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 Figure 4.(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 $w_{jk}$ is also nonzero.

**Specific explanation:**

- **Figure 3(a):** predecessor’s ($e_{12}$) has superior control over its successors ($e_{23}$ and $e_{24}$);
- **Figure 3(b):** design switches $s_{12}, s_{23}$ and $s_{24}$ to determine "on or off" of the edge;
- **Figure 3(d):** prioritize zero operation over other non-zero operations by adding one more node $i'$ between node $i$ and $j$. 

*Figure 3. An illustration for dependency.*
• How to apply Bayesian learning search strategy?

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

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

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

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

• 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

Figure 4. 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

Experiment:

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

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

less search time
• **Transferability to ImageNet:**

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

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