BANANAS: Bayesian Optimization with Neural Architectures for Neural Architecture Search

Colin White
Abacus.AI

Willie Neiswanger
Stanford University and Petuum, Inc.

Yash Savani
Abacus.AI
Neural architecture search

Neural architectures are getting increasingly more specialized and complex

Source: https://towardsdatascience.com/understanding-and-visualizing-densenets-7f688092391a
Roadmap

- Background
- “BayesOpt + neural predictor” framework
  - Encodings
  - Predictor / Uncertainty Calibration
  - Acquisition function
  - Acquisition function optimization

⇒ BANANAS
Cell-based search spaces

Search over a small labeled DAG

Stack the DAG on itself multiple times

- NAS-Bench-101 [Ying et al. 2019]
- NAS-Bench-201 [Dong & Yang 2019]
- DARTS [Liu et al. 2018]
Bayesian optimization

- NASBOT [Kandasamy et al. ‘18], Auto-Keras [Jin et al. ‘18]
- Popular method in HPO, but not straightforward for NAS
  - Gaussian process - scalability
  - Hand-designed distance function

“BO + Neural Predictor” Framework

- NASGBO [Ma et al. ‘19], BONAS [Shi et al. ‘19], BANANAS

Algorithm 1 BANANAS

**Input:** Search space $A$, dataset $D$, parameters $t_0$, $T$, $M$, $c$, $x$, acquisition function $\phi$, function $f(a)$ returning validation error of $a$ after training.
1. Draw $t_0$ architectures $a_0, \ldots, a_{t_0}$ uniformly at random from $A$ and train them on $D$.
2. For $t$ from $t_0$ to $T$,
   i. **Train an ensemble of meta neural networks** on $\{(a_0, f(a_0)), \ldots, (a_t, f(a_t))\}$.
   ii. **Generate a set of $c$ candidate architectures** from $A$ by randomly mutating the $x$ architectures $a$ from $\{a_0, \ldots, a_t\}$ that have the lowest value of $f(a)$.
   iii. For each candidate architecture $a$, evaluate the acquisition function $\phi(a)$.
   iv. Denote $a_{t+1}$ as the candidate architecture with minimum $\phi(a)$, and evaluate $f(a_{t+1})$.

**Output:** $a^* = \text{argmin}_{t=0,\ldots,T} f(a_t)$.
**Algorithm 1 BANANAS**

*Input*: Search space $A$, dataset $D$, parameters $t_0$, $T$, $M$, $c$, $x$, acquisition function $\phi$, function $f(a)$ returning validation error of $a$ after training.

1. Draw $t_0$ architectures $a_0, \ldots, a_{t_0}$ uniformly at random from $A$ and train them on $D$.
2. For $t$ from $t_0$ to $T$,
   i. Train an ensemble of meta neural networks on $\{(a_0, f(a_0)), \ldots, (a_t, f(a_t))\}$.
   ii. Generate a set of $c$ candidate architectures from $A$ by randomly mutating the $x$ architectures $a$ from $\{a_0, \ldots, a_t\}$ that have the lowest value of $f(a)$.
   iii. For each candidate architecture $a$, evaluate the acquisition function $\phi(a)$.
   iv. Denote $a_{t+1}$ as the candidate architecture with minimum $\phi(a)$, and evaluate $f(a_{t+1})$.

*Output*: $a^* = \min_{t=0, \ldots, T} f(a_t)$.

- Architecture encoding
- Uncertainty calibration
- Neural predictor architecture
- Acquisition optimization strategy
- Acquisition function
Most NAS algorithms use the adjacency matrix encoding.

Features are highly dependent on one another.
Path Encoding

Each path from input to output is a feature

Much more direct correlation with accuracy
Truncated Path Encoding

Exponential in the number of nodes

$3^0 + 3^1 + 3^2 + 3^3 + 3^4 + 3^5 = 364$
Theorem 4.1 (informal). Given integers $r, c > 0$, there exists an $N$ such that $\forall n > N$, there exists a set of $n$ paths $\mathcal{P}'$ such that the probability that $G_{n, n+c, r}$ contains a path not in $\mathcal{P}'$ is less than $\frac{1}{n^2}$.
Uncertainty prediction + architecture

GraphNN and path-encoding perform best
Acquisition Function

- Exploration vs. exploitation
Acquisition Function Optimization

- *Small mutations* of the best architectures is best
- Predictions are most accurate when close to training data
Exhaustive experiment

Path encoding; ITS; Mutation
**Algorithm 1 BANANAS**

**Input:** Search space $A$, dataset $D$, parameters $t_0$, $T$, $M$, $c$, $x$, acquisition function $\phi$, function $f(a)$ returning validation error of $a$ after training.

1. Draw $t_0$ architectures $a_0, \ldots, a_{t_0}$ uniformly at random from $A$ and train them on $D$.
2. For $t$ from $t_0$ to $T$,
   i. **Train an ensemble of meta neural networks** on $\{(a_0, f(a_0)), \ldots, (a_t, f(a_t))\}$.
   ii. **Generate a set of $c$ candidate architectures** from $A$ by randomly mutating the $x$ architectures $a$ from $\{a_0, \ldots, a_t\}$ that have the lowest value of $f(a)$.
   iii. For each candidate architecture $a$, evaluate the acquisition function $\phi(a)$.
   iv. Denote $a_{t+1}$ as the candidate architecture with minimum $\phi(a)$, and evaluate $f(a_{t+1})$.

**Output:** $a^* = \arg\min_{t=0, \ldots, T} f(a_t)$. 

---

**Path encoding, ensemble**

**Small mutations**

**Independent Thompson Sampling**
NASBench-101 and DARTS Results

Table 1: Comparison of NAS algorithms on the DARTS search space. The runtime unit is total GPU-days on a Tesla V100.

<table>
<thead>
<tr>
<th>NAS Algorithm</th>
<th>Source</th>
<th>Avg. Test error</th>
<th>Runtime</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random search</td>
<td>[35]</td>
<td>3.29</td>
<td>4</td>
<td>Random</td>
</tr>
<tr>
<td>Local search</td>
<td>[66]</td>
<td>3.49</td>
<td>11.8</td>
<td>Local search</td>
</tr>
<tr>
<td>DARTS</td>
<td>[35]</td>
<td>2.76</td>
<td>5</td>
<td>Gradient-based</td>
</tr>
<tr>
<td>ASHA</td>
<td>[30]</td>
<td>3.03</td>
<td>9</td>
<td>Successive halving</td>
</tr>
<tr>
<td>DARTS</td>
<td>Ours</td>
<td>2.68</td>
<td>5</td>
<td>Gradient-based</td>
</tr>
<tr>
<td>ASHA</td>
<td>Ours</td>
<td>3.08</td>
<td>9</td>
<td>Successive halving</td>
</tr>
<tr>
<td>BANANAS</td>
<td>Ours</td>
<td><strong>2.64</strong></td>
<td>11.8</td>
<td>BO + neural predictor</td>
</tr>
</tbody>
</table>
NASBench-201 Results
## Subsequent Work

<table>
<thead>
<tr>
<th>NAS Methods</th>
<th>#Queries</th>
<th>Test Accuracy (%)</th>
<th>Encoding</th>
<th>Search Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random Search [23]</td>
<td>1000</td>
<td>93.54</td>
<td>Discrete</td>
<td>Random</td>
</tr>
<tr>
<td>RL [23]</td>
<td>1000</td>
<td>93.58</td>
<td>Discrete</td>
<td>REINFORCE</td>
</tr>
<tr>
<td>BO [23]</td>
<td>1000</td>
<td>93.72</td>
<td>Discrete</td>
<td>Bayesian Optimization</td>
</tr>
<tr>
<td>RE [23]</td>
<td>1000</td>
<td>93.72</td>
<td>Discrete</td>
<td>Evolution</td>
</tr>
<tr>
<td>NAO [14]</td>
<td>1000</td>
<td>93.74</td>
<td>Supervised</td>
<td>Gradient Decent</td>
</tr>
<tr>
<td>BANANAS [49]</td>
<td>500</td>
<td>94.08</td>
<td>Supervised</td>
<td>Bayesian Optimization</td>
</tr>
<tr>
<td>RL (ours)</td>
<td>400</td>
<td>93.74</td>
<td>Supervised</td>
<td>REINFORCE</td>
</tr>
<tr>
<td>BO (ours)</td>
<td>400</td>
<td>93.79</td>
<td>Supervised</td>
<td>Bayesian Optimization</td>
</tr>
<tr>
<td>arch2vec-RL</td>
<td>400</td>
<td><strong>94.10</strong></td>
<td>Unsupervised</td>
<td>REINFORCE</td>
</tr>
<tr>
<td>arch2vec-BO</td>
<td>400</td>
<td>94.05</td>
<td>Unsupervised</td>
<td>Bayesian Optimization</td>
</tr>
</tbody>
</table>

[Yan et al. ‘20]

![Graph of test error vs. encoding length]

[White et al. ‘20]

### True Benchmark

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Test Error (in %)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TPE</td>
<td>6.43 ± 0.16</td>
</tr>
<tr>
<td>BOHB</td>
<td>6.40 ± 0.12</td>
</tr>
<tr>
<td>Random Search</td>
<td>6.36 ± 0.12</td>
</tr>
<tr>
<td>Alpha X</td>
<td>6.31 ± 0.13</td>
</tr>
<tr>
<td>NASBOT</td>
<td>6.35 ± 0.10</td>
</tr>
<tr>
<td>Reg Evolution</td>
<td>6.20 ± 0.13</td>
</tr>
<tr>
<td>ReMAADE</td>
<td>6.15 ± 0.13</td>
</tr>
<tr>
<td>BANANAS</td>
<td>5.77 ± 0.31</td>
</tr>
</tbody>
</table>

[Krishna et al. ‘20]
Conclusion

- “BO + Neural Predictor” is a powerful NAS framework
  - Encoding, surrogate model, acquisition function, acquisition function optimization
- BANANAS is a performant instantiation of the framework

https://github.com/naszilla/naszilla

Thanks!