SMASH: One-Shot Model Architecture
Search Through HyperNetworks

Authors: Andrew Brock, Theodore Lim, J.M. Ritchie, and Nick Weston
April 14, 2018

Presentation by Kamal Rai
When training neural networks, we:

- Fix the network architecture
- Specify a loss function $L$
- Find optimal weights $W$ using backprop to minimize $\frac{dL}{dW}$

Iterate over design decisions until we obtain a good model.
Model hyperparameters: Depth, width, connectivity
Finding optimal architectures requires extensive experimentation. Current automated architecture selection methods are expensive. Evolutionary techniques and reinforcement learning.

Given randomly sampled hyperparameters $c$, we can iteratively:

1. Optimize the weights of an auxiliary network using $\frac{\partial L(w_c)}{\partial W_c} \frac{\partial W_c}{\partial c}$
2. Optimize the weights of the main network
Figure 1: Generate weights using an auxiliary network
Algorithm 1: **SMASH**

**Input:** Space of all candidate architectures $\mathbb{R}_c$

Initialize HyperNet weights $H$

loop

Sample input minibatch $x_i$, random architecture $c$, and architecture weights $W(c)$

Get training error $E_t = f_c(W, x_i) = f_c(H(c), x_i)$, backprop $\frac{dE}{dW}$ through the HyperNet and then update $H$

end loop

loop

Sample a random architecture $c$ and evaluate error on validation set $E_v = f_c(H(c), x_v)$

end loop

Fix architecture and train normally with freely-varying weights $W$
Figure 2: Sampling from a hypernetwork
Ranking Candidate Models

Figure 3: Exploring performance on CIFAR-100
The strength of correlation depends on

- The capacity of the hypernet
- The ratio of hypernet generated weights to freely learned weights
Figure 4: Layers are ops that read and write to memory
Table 1: Error rates (%) on CIFAR-10 and CIFAR-100 with standard data augmentation (+).

<table>
<thead>
<tr>
<th>Method</th>
<th>Depth</th>
<th>Params</th>
<th>C10+</th>
<th>C100+</th>
</tr>
</thead>
<tbody>
<tr>
<td>FractalNet [20]</td>
<td>21</td>
<td>38.6M</td>
<td>5.22</td>
<td>23.30</td>
</tr>
<tr>
<td>with Dropout/Drop-path</td>
<td>21</td>
<td>38.6M</td>
<td>4.60</td>
<td>23.73</td>
</tr>
<tr>
<td>Wide ResNet [43]</td>
<td>16</td>
<td>11.0M</td>
<td>4.81</td>
<td>22.07</td>
</tr>
<tr>
<td></td>
<td>28</td>
<td>36.5M</td>
<td>4.17</td>
<td>20.50</td>
</tr>
<tr>
<td>DenseNet-BC (k = 24) [15]</td>
<td>250</td>
<td>15.3M</td>
<td>3.62</td>
<td>17.60</td>
</tr>
<tr>
<td>DenseNet-BC (k = 40)</td>
<td>190</td>
<td>25.6M</td>
<td>3.46</td>
<td>17.18</td>
</tr>
<tr>
<td>Neural Architecture Search w/ RL[44]</td>
<td>39</td>
<td>32.0M</td>
<td>3.84</td>
<td>-</td>
</tr>
<tr>
<td>Large-Scale Evolution [26]</td>
<td>-</td>
<td>5.4M</td>
<td>5.40</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>40.4M</td>
<td>-</td>
<td>23.7</td>
</tr>
<tr>
<td>CGP-CNN [38]</td>
<td>-</td>
<td>1.68M</td>
<td>5.98</td>
<td>-</td>
</tr>
<tr>
<td>SMASHv1</td>
<td>116</td>
<td>4.6M</td>
<td>5.53</td>
<td>22.07</td>
</tr>
<tr>
<td>SMASHv2</td>
<td>211</td>
<td>16M</td>
<td>4.03</td>
<td>20.60</td>
</tr>
</tbody>
</table>

**Figure 5:** Benchmark results
Limitations

- The space of candidate architectures must be pre-specified
- Does not address regularization or learning rate
- Not jointly training the hypernet and the main network
- Not using gradients to optimize the choice of main network
Conclusion

Can efficiently explore architectures using Hypermnet weights

Two Related Works

- Hyperparameter Optimization with Hypermets. J. Lorraine and D. Duvenaud