Averaging Weights Leads to Wider Optima and Better Generalization

Pavel Izmailov\textsuperscript{1} \hspace{1cm} Dmitrii Podoprikhin\textsuperscript{2,3} \hspace{1cm} Timur Garipov\textsuperscript{4,5}
Dmitry Vetrov\textsuperscript{2,3} \hspace{1cm} Andrew Gordon Wilson\textsuperscript{1}

\textsuperscript{1}Cornell University
\textsuperscript{2}Higher School of Economics
\textsuperscript{3}Samsung-HSE Laboratory
\textsuperscript{4}Samsung AI Center in Moscow
\textsuperscript{5}Lomonosov Moscow State University

Uncertainty in Artificial Intelligence
Monterey, California, USA

August 9, 2018
Optima width is conjectured to be correlated with generalization (Keskar et al. [2017], Hochreiter and Schmidhuber [1997])
Talk Outline

We propose Stochastic Weight Averaging (SWA) — an equally weighted running average of parameters (DNN weights) traversed by SGD with a modified learning (cyclical or high constant) rate schedule.

▶ Improves generalization
▶ No significant computational overhead
▶ Extremely easy to implement and use

Explanation:
▶ Finds wider solutions centered in the set of high-performing networks
▶ Approximates ensembling
SGD Experiment: Constant Learning Rate

Run SGD with constant learning rate and visualize trajectory

» SGD iterates stay at the boundary of a high-quality region
» Averaging iterates improves performance
» Shift between train and test
Explanation: Soap Bubble

Mandt et al. [2017]: SGD with fixed learning rate samples from a Gaussian distribution centered at the minimum of the loss.

SGD iterates concentrate on a surface of an ellipsoid. Averaging lets us go inside the ellipsoid!
What if we use a cyclical learning rate?
SGD Experiment: Cyclical Learning Rate

Observations still hold:

- SGD iterates stay at the boundary of a high-quality region
- Averaging iterates improves performance
- Shift between train and test
Explanation: Ensemble Approximation

- SGD is taking small steps, so averaging weights $\approx$ ensembling by linearization

$$f \left( \frac{1}{n} \sum_{i=1}^{n} w_i \right) \approx \frac{1}{n} \sum_{i=1}^{n} f(w_i)$$

- Empirically, averaging weights and ensembling SGD iterates indeed lead to similar predictions
SWA details

- Use learning rate schedule that doesn’t decay to zero (cyclical or constant)
- Average weights
  - Cyclical LR ⇒ at the end of each cycle
  - Constant LR ⇒ at the end of each epoch
- Recompute Batch Normalization statistics at the end of training; in practice do one additional forward pass on train data
SWA details

- Use learning rate schedule that doesn’t decay to zero (cyclical or constant)
- Average weights
  - Cyclical LR ⇒ at the end of each cycle
  - Constant LR ⇒ at the end of each epoch
- Recompute Batch Normalization statistics at the end of training; in practice do one additional forward pass on train data
SWA vs SGD

Run SGD and SWA from the same initialization (ResNet-164, CIFAR-100)

- SGD achieves better train loss
- SWA achieves better test accuracy
- Large shift between train and test
Connecting SWA and SGD Solutions

\[ w(t) = t \cdot w_{\text{SGD}} + (1 - t) \cdot w_{\text{SWA}} \]
Width along random rays

$$w(t) = \{w_{\text{SWA}}, w_{\text{SGD}}\} + t \cdot \frac{d}{\|d\|}, \quad d \sim \mathcal{N}(0, I)$$
Width along random rays

\[ w(t) = \{w_{\text{SWA}}, w_{\text{SGD}}\} + t \cdot \frac{d}{\|d\|}, \quad d \sim \mathcal{N}(0, I) \]
## SWA Results

<table>
<thead>
<tr>
<th>DNN (Budget)</th>
<th>SGD</th>
<th>SWA 1 Budget</th>
<th>SWA 1.5 Budget</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIFAR-100</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VGG-16 (200)</td>
<td>72.55 ± 0.10</td>
<td>73.91 ± 0.12</td>
<td>74.27 ± 0.25</td>
</tr>
<tr>
<td>ResNet-164 (150)</td>
<td>78.49 ± 0.36</td>
<td>79.77 ± 0.17</td>
<td>80.35 ± 0.16</td>
</tr>
<tr>
<td>WRN-28-10 (200)</td>
<td>80.82 ± 0.23</td>
<td>81.46 ± 0.23</td>
<td>82.15 ± 0.27</td>
</tr>
<tr>
<td>PyramidNet-272 (300)</td>
<td>83.41 ± 0.21</td>
<td>–</td>
<td>84.16 ± 0.15</td>
</tr>
<tr>
<td>CIFAR-10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VGG-16 (200)</td>
<td>93.25 ± 0.16</td>
<td>93.59 ± 0.16</td>
<td>93.64 ± 0.18</td>
</tr>
<tr>
<td>ResNet-164 (150)</td>
<td>95.28 ± 0.10</td>
<td>95.56 ± 0.11</td>
<td>95.83 ± 0.03</td>
</tr>
<tr>
<td>WRN-28-10 (200)</td>
<td>96.18 ± 0.11</td>
<td>96.45 ± 0.11</td>
<td>96.79 ± 0.05</td>
</tr>
<tr>
<td>ShakeShake-2x64d (1800)</td>
<td>96.93 ± 0.10</td>
<td>–</td>
<td>97.12 ± 0.06</td>
</tr>
<tr>
<td>Imagenet</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DNN</td>
<td>SGD</td>
<td>SWA 5 epochs</td>
<td>SWA 10 epochs</td>
</tr>
<tr>
<td>ResNet-50</td>
<td>76.15</td>
<td>76.83 ± 0.01</td>
<td>76.97 ± 0.05</td>
</tr>
<tr>
<td>ResNet-152</td>
<td>78.31</td>
<td>78.82 ± 0.01</td>
<td>78.94 ± 0.07</td>
</tr>
<tr>
<td>DenseNet-161</td>
<td>77.65</td>
<td>78.26 ± 0.09</td>
<td>78.44 ± 0.06</td>
</tr>
</tbody>
</table>
Applications and Extensions

- Two papers at UDL workshop tomorrow!
  - Improving Stability in Deep Reinforcement Learning with Weight Averaging
  - Fast Uncertainty Estimates and Bayesian Model Averaging of DNNs
- Athiwaratkun et al. [2018]: use a modified version of SWA to get SOTA results in Semi-Supervised Learning
Summary

- SWA is a simple technique that consistently improves generalization with deep neural networks with virtually no computational overhead.
- SWA is very easy to use and implement and proved useful in a range of applications.
- Code is available, so we encourage you to try SWA for yourself!
  - PyTorch: https://github.com/timgaripov/swa
  - Chainer: https://github.com/chainer/models/tree/master/swa
  - fast.ai: https://github.com/fastai/fastai
References


