Motivation

- Hyperparameters such as architecture choice, data augmentation, and dropout are crucial for neural net generalization, but difficult to tune.
- Grid search, random search, and Bayesian optimization treat hyperparameter tuning as a black-box problem, which does not scale to high-dimensional hyperparameters.
- Hyperparameter tuning is a bilevel optimization problem: $\lambda^* = \arg\min_{\lambda} \mathcal{L}_V(\lambda, \mathbf{w}^*(\lambda))$ s.t. $\mathbf{w}^*(\lambda) = \arg\min_{\mathbf{w}} \mathcal{L}_T(\lambda, \mathbf{w})$.
- We approximate the best-response function $\mathbf{w}^*(\lambda)$ with a hypernetwork $\mathbf{w}_s(\lambda)$, called a Self-Tuning Network (STN).

Summary

- We propose a compact architecture for approximating neural net best-responses, that can be used as a drop-in replacement for existing deep learning modules.
- Our training algorithm alternates between approximating the best-response around the current hyperparameters and optimizing the hyperparameters with the approximate best-response.
- This yields a gradient-based algorithm that is (1) computationally inexpensive, (2) can optimize all regularization hyperparameters including discrete hyperparameters, and (3) scales to large NNs.
- Our approach discovers hyperparameter trajectories that can outperform fixed hyperparameter values.

Self-Tuning Network (STN) Training Algorithm

Initialize: Hypernetwork parameters $\phi$, hyperparameters $\lambda$ 
while not converged do
  for $t = 1, \ldots, T_{\text{train}}$ do
    $\epsilon \sim p(\epsilon | \sigma)$
    $\phi \leftarrow \phi - \alpha_{\phi} \nabla_{\phi} \mathcal{L}_T(\lambda + \epsilon, \mathbf{w}_s(\lambda + \epsilon))$
  for $t = 1, \ldots, T_{\text{valid}}$ do
    $\epsilon \sim p(\epsilon | \sigma)$
    $\mathcal{L}_V(\lambda, \sigma) \leftarrow \mathcal{L}_V(\lambda + \epsilon, \mathbf{w}_s(\lambda + \epsilon)) - \tau \mathbb{E}[p(\epsilon | \sigma)]$
    $\lambda, \sigma \leftarrow (\lambda, \sigma) - \alpha_{\lambda,\sigma} \nabla_{\lambda,\sigma} \mathcal{L}_V(\lambda, \sigma)$

• We scale and shift the network’s hidden units (i.e., the rows of weights and biases) by an amount which depends on our hyperparameters:

$$W_{o}(\lambda) = W_{\text{shift}} + (U \lambda) \odot \text{softmax} \ W_{\text{scale}}$$
$$b_{o}(\lambda) = b_{\text{shift}} + (C \lambda) \odot \text{softmax} \ b_{\text{scale}}$$

• Memory efficient (roughly $2x$ no. of parameters) and scales well to high dimensions.

Sampling Hyperparameters

Legend:
- Exact best response $\phi_{o}^*(\lambda)$
- Approximate best response $\phi_{o}^{approx}(\lambda)$
- Hyperparameter distribution $p(\epsilon | \sigma)$

Hyperparameter Trajectories

- STNs discover hyperparameter trajectories which can outperform fixed hyperparameters.
- For a single dropout rate, STNs implement a curriculum with a gradually increasing dropout probability.
- The same trajectory is followed regardless of the initial hyperparameter value.

Real-World Datasets

<table>
<thead>
<tr>
<th>Method</th>
<th>PTB Val Perplexity</th>
<th>PTB Test Perplexity</th>
<th>CUB Val Loss</th>
<th>CUB Test Loss</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grid Search</td>
<td>97.32</td>
<td>94.58</td>
<td>0.794</td>
<td>0.809</td>
</tr>
<tr>
<td>Random Search</td>
<td>84.81</td>
<td>81.46</td>
<td>0.921</td>
<td>0.752</td>
</tr>
<tr>
<td>Bayesian Optimizer</td>
<td>72.13</td>
<td>69.29</td>
<td>0.636</td>
<td>0.651</td>
</tr>
<tr>
<td>STN (Ours)</td>
<td>70.30</td>
<td>67.68</td>
<td>0.576</td>
<td>0.576</td>
</tr>
</tbody>
</table>

Final validation/test performance on PTB and CIFAR-10

• CNN time comparison
• LSTM time comparison
• Hyperparameter trajectories for LSTM tuning