Self-Tuning Networks for Hyperparameter Optimization

Matthew MacKay*, Paul Vicol*, Jon Lorraine, David Duvenaud, Roger Grosse

University of Toronto & Vector Institute

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:

$$\boldsymbol{\lambda}^* = \arg\min_{\boldsymbol{\lambda}} \mathcal{L}_V(\boldsymbol{\lambda}, \mathbf{w}^*(\boldsymbol{\lambda})) \quad \text{s.t.} \quad \mathbf{w}^*(\boldsymbol{\lambda}) = \arg\min_{\mathbf{w}} \mathcal{L}_T(\boldsymbol{\lambda}, \mathbf{w})$$

- We approximate the best-response function $\mathbf{w}^*(\boldsymbol{\lambda})$ with a hypernetwork $\mathbf{w}_\phi(\boldsymbol{\lambda})$, called a Self-Tuning Network (STN).

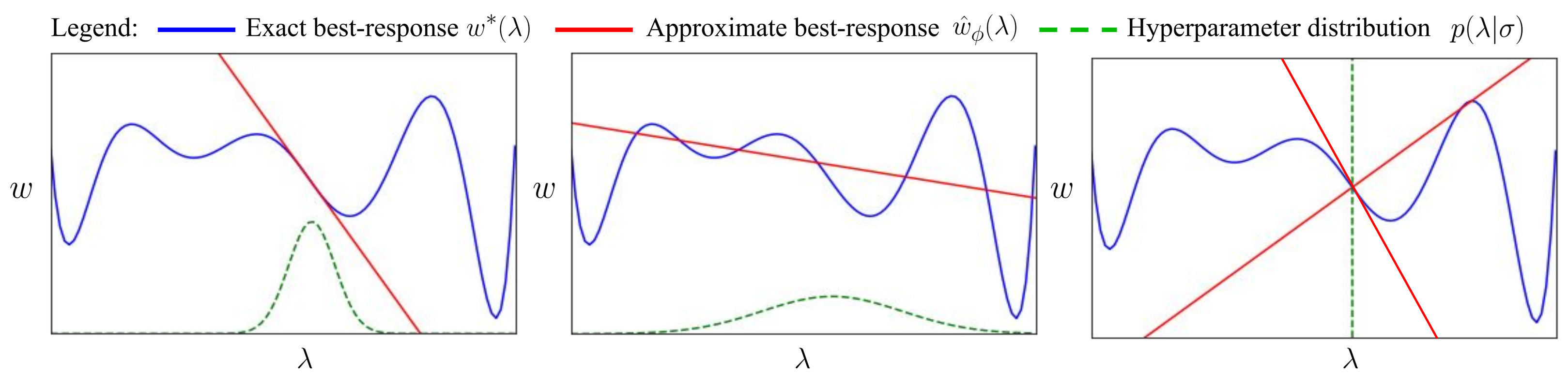
Summary

- 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

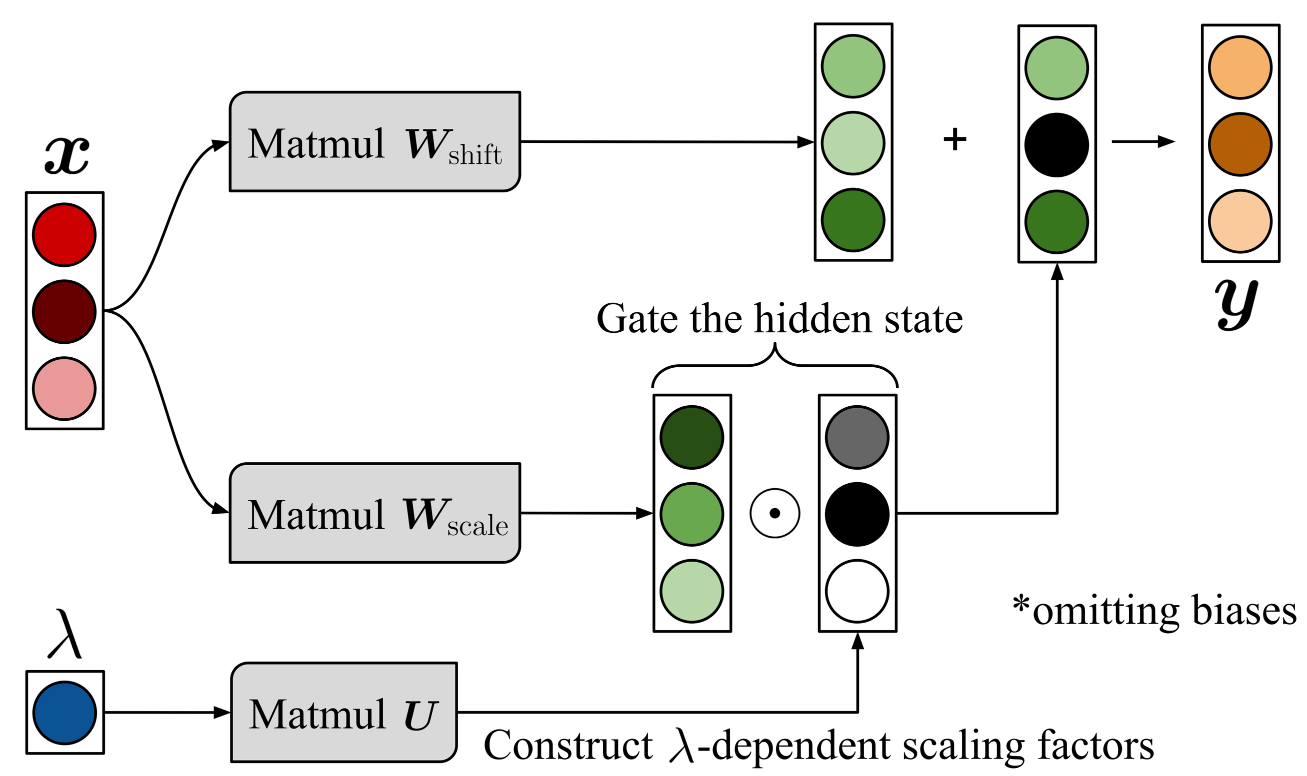
Initialize: Hypernetwork parameters ϕ, hyperparameters $\boldsymbol{\lambda}$
while not converged **do**
  for $t = 1, \ldots, T_{train}$ **do**
    $\epsilon \sim p(\epsilon|\boldsymbol{\sigma})$
    $\phi \leftarrow \phi - \alpha_1 \nabla_\phi \left[\mathcal{L}_T(\boldsymbol{\lambda} + \epsilon, \hat{\mathbf{w}}_\phi(\boldsymbol{\lambda} + \epsilon))\right]$
  for $t = 1, \ldots, T_{valid}$ **do**
    $\epsilon \sim p(\epsilon|\boldsymbol{\sigma})$
    $\hat{\mathcal{L}}_V(\boldsymbol{\lambda}, \boldsymbol{\sigma}) \leftarrow \mathcal{L}_V(\boldsymbol{\lambda} + \epsilon, \hat{\mathbf{w}}_\phi(\boldsymbol{\lambda} + \epsilon)) - \tau \mathbb{H}[p(\epsilon|\boldsymbol{\sigma})]$
    $(\boldsymbol{\lambda}, \boldsymbol{\sigma}) \leftarrow (\boldsymbol{\lambda}, \boldsymbol{\sigma}) - \alpha_2 \nabla_{\boldsymbol{\lambda}, \boldsymbol{\sigma}} \left[\hat{\mathcal{L}}_V(\boldsymbol{\lambda}, \boldsymbol{\sigma})\right]$

Sampling Hyperparameters



- *Just right* → the gradient of the approximation will match that of the best-response.
- *Too wide* → the hypernetwork may be insufficiently flexible to model the best-response, and the gradients will not match.
- *Too small* → the hypernetwork will match the best-response at the current hyperparameter, but may not be locally correct.
- We re-parameterize the hyperparameter $\boldsymbol{\lambda}$ to lie in $\mathbb{R}$ and use noise distribution $p(\epsilon|\boldsymbol{\sigma}) = \mathcal{N}(0, \boldsymbol{\sigma})$.

Hypernetwork Best-Response Architecture



- We scale and shift the network's hidden units (≡ the rows of weights and biases) by an amount which depends on our hyperparameters:

$$\hat{W}_\phi(\boldsymbol{\lambda}) = W_{\text{shift}} + (U\boldsymbol{\lambda}) \odot_{\text{row}} W_{\text{scale}}$$

$$\hat{b}_\phi(\boldsymbol{\lambda}) = b_{\text{shift}} + (C\boldsymbol{\lambda}) \odot_{\text{row}} b_{\text{scale}}$$

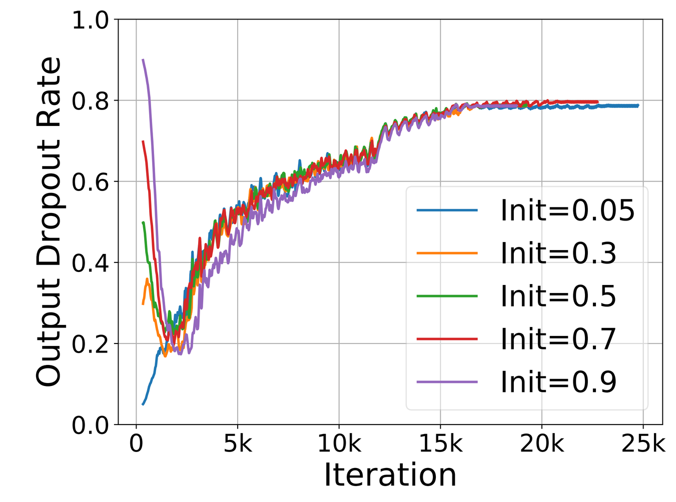
- Memory efficient (roughly 2x no. of parameters) and scales well to high dimensions.

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.

Method	Val	Test
$p = 0.68$ (Fixed)	85.83	83.19
$p \sim \mathcal{N}(0.68, \sigma = 0.05)$	85.87	82.29
$p = 0.68 + 0.1\sin(k\pi)$	85.29	82.15
$p = 0.78$ (Converged STN)	89.65	86.90
STN (Ours)	**82.58**	**79.02**
Following STN Trajectory	82.87	79.93

Comparing hyperparameter trajectories

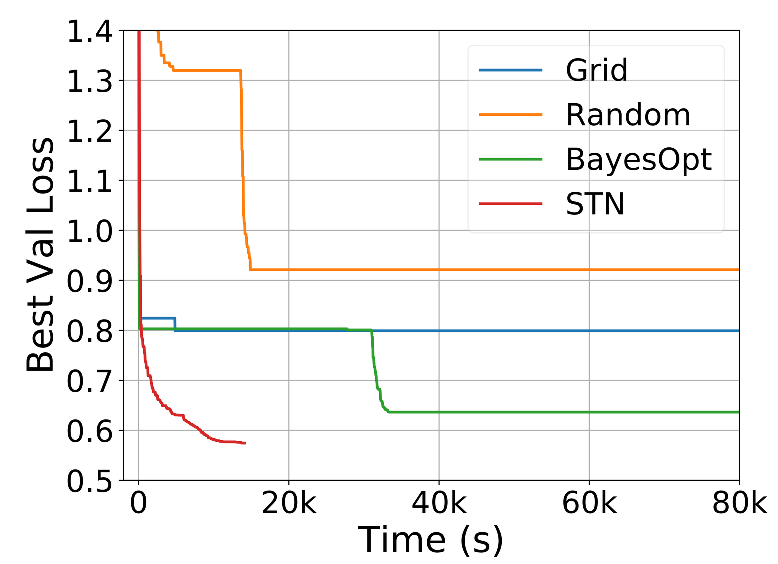


Real-World Datasets

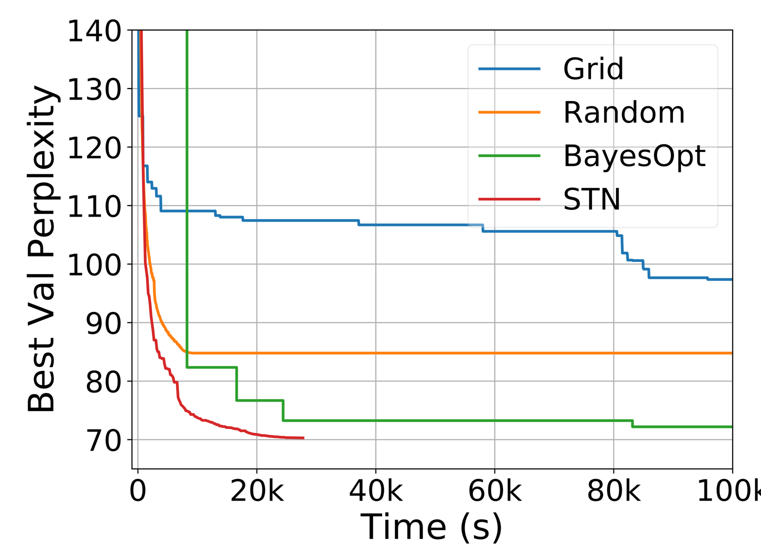
Method	PTB Val Perplexity	PTB Test Perplexity	CIFAR-10 Val Loss	CIFAR-10 Test Loss
Grid Search	97.32	94.58	0.794	0.809
Random Search	84.81	81.46	0.921	0.752
Bayesian Optimization	72.13	69.29	0.636	0.651
STN	**70.30**	**67.68**	**0.575**	**0.576**

Final validation/test performance on PTB and CIFAR-10

- CNN time comparison



- LSTM time comparison



- Hyperparameter trajectories for LSTM tuning