Δ-STN: Efficient Bilevel Optimization for Neural Networks using Structured Response Jacobians

Juhan Bae, Roger Grosse

University of Toronto, Vector Institute

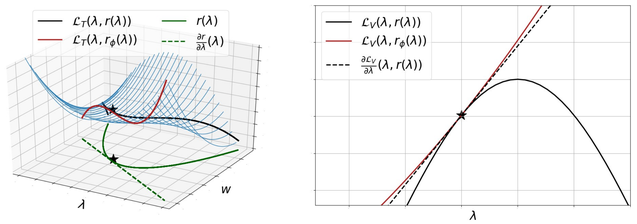
Hyperparameter Optimization

Hyperparameter optimization can be formulated as a bilevel optimization problem.

$$\min_{\lambda} \mathcal{L}_V(\lambda, w^*) \text{ subject to } w^* = \arg\min_{w} \mathcal{L}_T(\lambda, w)$$

Goal: Find the optimal hyperparameters λ^* that minimizes the validation objective after training.

Parametric Best-Response Function



Best-response function maps hyperparameters to the optimal parameters w^*:

$$r(\lambda) = \arg\min_{w} \mathcal{L}_T(\lambda, w)$$

Substituting the best-response function in the validation objective converts the bilevel problem to a single-level problem.

Idea: Learn a **parametric best-response function** r_ϕ with hypernetwork and **jointly** optimize hypernetwork and hyperparameters.

$$\min_{\phi} \mathbb{E}_{\varepsilon \sim p(\varepsilon|\sigma)} \left[\mathcal{L}_T(\lambda + \varepsilon, r_\phi(\lambda + \varepsilon)) \right], \quad \min_{\lambda} \mathcal{L}_V(\lambda, r_\phi(\lambda))$$

Self-Tuning Networks (STNs)

Self-Tuning Networks (STNs) parameterize the best-response functions (with further structure imposed) as:

$$r_\phi(\lambda) = \Phi\lambda + \phi_0$$

Method: Jointly optimize hypernetwork and hyperparameters with **structured** best-response function.

Contributions

Introduce an improved hypernetwork architecture that efficiently optimizes hyperparameters online.

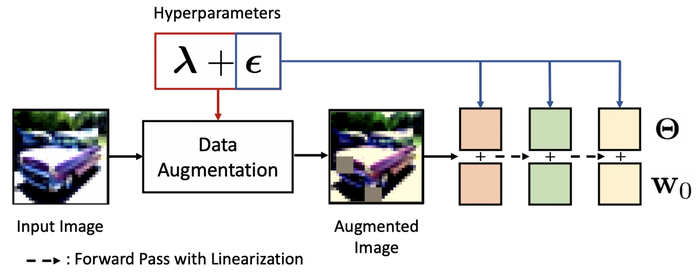
- Improve convergence and stability in training Self-Tuning Networks (STNs).
- Accurate approximation of the best-response Jacobian instead of the full best-response function.
- Achieve better generalization in less time, compared to existing approaches to bilevel optimization.
- Open-sourced (Δ-)Self-Tuning Networks: `https://github.com/pomonam/Self-Tuning-Networks`

Fix 1: Centered Hypernetwork

Use a **centered parameterization** of the hypernetwork (with further structure imposed for neural networks):

$$r_\theta(\lambda, \lambda_0) = \Theta(\lambda - \lambda_0) + w_0$$

- λ_0: "current hyperparameters", w_0: "current weights"
- Θ: how the weights are adjusted in response to a perturbation to λ (**best-response Jacobian**)



Use a **modified update rule** to separately train parameters and best-response Jacobian.

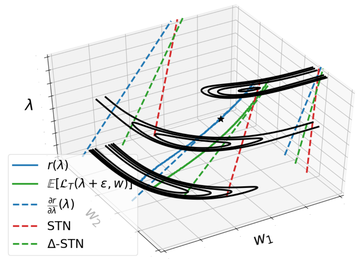
$$\arg\min_{w_0} \mathcal{L}_T(\lambda, w_0), \quad \arg\min_{\Theta} \mathbb{E}_{\varepsilon \sim p(\varepsilon|\sigma)} \left[\mathcal{L}_T(\lambda + \varepsilon, r_\theta(\lambda + \varepsilon, \lambda)) \right]$$

Advantages:

- Improve the conditioning of the Gauss-Newton Hessian.
- Fix undesirable bilevel optimization dynamics.
- Eliminate any bias to the optimal w_0^* induced by the perturbation.
- Enhance the stability and convergence in training STNs.

Fix 2: Direct Approximation of Best-Response Jacobian

It is unnecessary to account for the **nonlinear effect** of large changes to w_0, as the hypernetwork is only used to estimate the best-response Jacobian.



Idea: Linearize the network around the current parameters $r(\lambda_0) = w_0$.

Directly approximate the best-response Jacobian instead of learning the full best-response function.

Network prediction with linearized hypernetwork:

$$y' = f(x, r_\theta(\lambda_0 + \varepsilon, \lambda_0), \lambda_0 + \varepsilon, \xi) \approx f(x, r_\theta(\lambda_0, \lambda_0), \lambda_0 + \varepsilon, \xi) + J_{yw}\Delta w$$
$$= f(x, w_0, \lambda_0 + \varepsilon, \xi) + J_{yw}\Theta\varepsilon,$$

where $J_{yw} = \partial y / \partial w$ is the weight-output Jacobian. This relationship can also be written with $\Delta y \approx J_{yw}\Delta w = J_{yw}\Theta\varepsilon$. The linearization can be efficiently computed using forward mode automatic differentiation

Advantage: Accurate approximation of the best-response Jacobian.

Training Algorithm for Δ-STNs

Method: Jointly optimize the **structured** hypernetwork and hyperparameters.

Initialize: hypernetwork $\theta = \{w_0, \Theta\}$, hyperparameters λ.
while *not converged* **do**
  for $t = 1, ..., T_{train}$ **do**
    $\varepsilon \sim p(\varepsilon|\sigma)$
    $w_0 \leftarrow w_0 - \alpha_1 \nabla_{w_0}(\mathcal{L}_T(\lambda, r_\theta(\lambda, \lambda)))$
    $\Theta \leftarrow \Theta - \alpha_1 \nabla_{\Theta}(\mathcal{L}_T(\lambda + \varepsilon, r_\theta(\lambda + \varepsilon, \lambda)))$ ▷ Linearization with fwd mode autodiff.
  for $t = 1, ..., T_{valid}$ **do**
    $\varepsilon \sim p(\varepsilon|\sigma)$
    $\lambda \leftarrow \lambda - \alpha_2 \nabla_{\lambda}(\mathcal{L}_V(\lambda + \varepsilon, r_\theta(\lambda + \varepsilon, \lambda_0)))|_{\lambda=\lambda_0}$
    $\sigma \leftarrow \sigma - \alpha_3 \nabla_{\sigma}(\mathcal{L}_V(\lambda + \varepsilon, r_\theta(\lambda + \varepsilon, \lambda)) - \tau\mathbb{H}[p(\varepsilon|\sigma)])$

Linear Regression on UCI dataset

Tune L^2 regularization and input dropout rate.

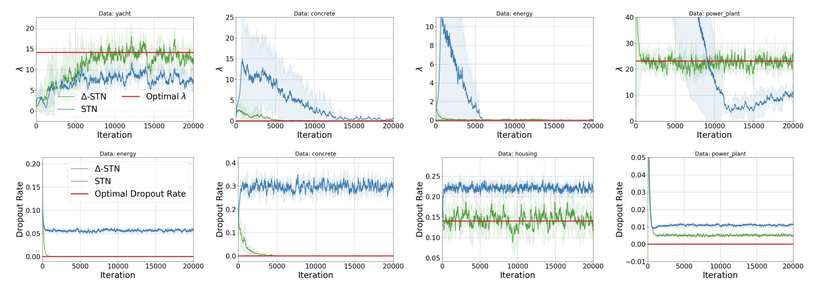


Image Classification & Language Modelling Tasks

Tune ∼30 regularization hyperparameters (e.g. dropout rate, number of Cutout holes, random translation/degrees) on various datasets and architectures.

Dataset	Network	RS	BO	STN	Centered STN	Δ-STN
MNIST	MLP	0.043 (0.042)	0.042 (0.043)	0.043 (0.041)	0.041 (0.039)	**0.040 (0.038)**
FMNIST	SimpleCNN	0.206 (0.214)	0.217 (0.215)	0.196 (0.218)	0.191 (0.212)	**0.189 (0.209)**
CIFAR10	AlexNet	0.631 (0.671)	0.594 (0.598)	0.474 (0.488)	0.431 (0.450)	**0.425 (0.446)**
	VGG16	0.566 (0.595)	0.421 (0.446)	0.330 (0.354)	0.286 (0.321)	**0.272 (0.296)**
	ResNet18	0.264 (0.298)	0.230 (0.267)	0.266 (0.312)	0.222 (0.258)	**0.204 (0.238)**
PTB	LSTM	84.81 (81.46)	72.13 (69.29)	70.67 (67.78)	69.40 (66.67)	**68.63 (66.26)**

Figure: Final validation/test performance on image classification and language modelling tasks.

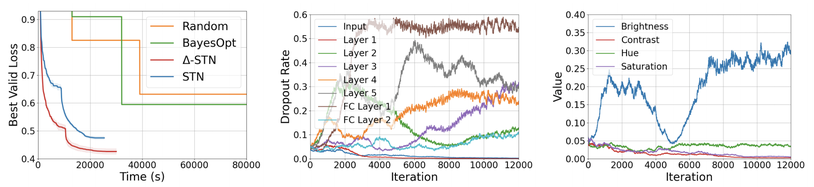


Figure: A comparison of the best validation loss achieved by random search, Bayesian optimization, STNs, and Δ-STNs over time for AlexNet. Hyperparameter schedules found by Δ-STNs.