Stochastic Hyperparameter Optimization with Hypernets

Jonathan Lorraine, David Duvenaud

University of Toronto

Main Idea

- Machine learning models often nest optimization of model weights in the optimization of hyperparameters.
- We collapse the nested optimization into joint optimization by training a neural network to output optimal weights for each hyperparameter.
- The method converges to locally optimal weights and hyperparameters for large hypernets and effectively tunes thousands of hyperparameters.

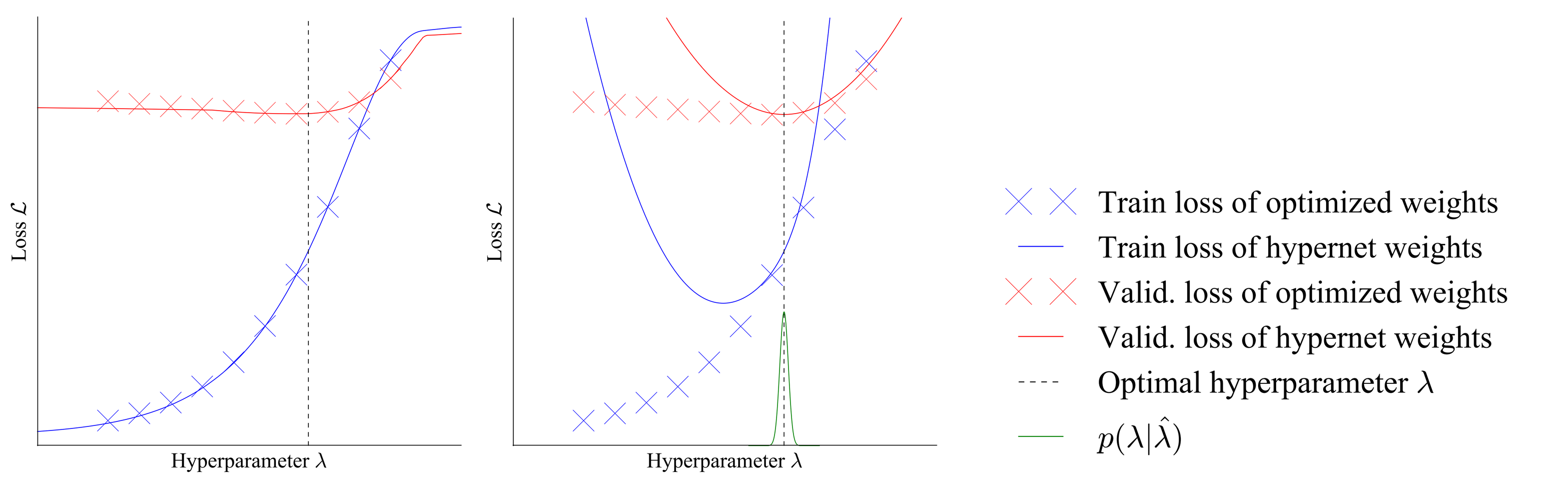


Figure 2: Training and validation loss of a neural net for linear regression on MNIST, estimated by cross-validation (crosses) or by a hypernet (lines), which outputs $7,850$-dimensional network weights. The training and validation loss can be cheaply evaluated at any hyperparameter value using a hypernet. Standard cross-validation requires training from scratch each time. *Left:* A global approximation the best-response. *Right:* A local approximation to the best-response.

Hyperparameter Tuning is Nested Optimization

- Selecting a hyperparameter is finding a solution to the following bi-level optimization problem:

$$\underset{\lambda}{\operatorname{argmin}} \underset{\text{Valid.}}{\mathcal{L}} \left(\underset{\mathrm{w}}{\operatorname{argmin}} \underset{\text{Train}}{\mathcal{L}} (\mathrm{w}, \lambda) \right) \tag{1}$$

- The optimized model weights depend on the choice of hyperparameter. This is a best-response function of the weights to the hyperparameters:

$$\mathrm{w}^*(\lambda) = \underset{\mathrm{w}}{\operatorname{argmin}} \underset{\text{Train}}{\mathcal{L}} (\mathrm{w}, \lambda) \tag{2}$$

Learning a Mapping from Hyperparameters to Optimal Weights

- A hypernet is a neural network which outputs network weights.
- The best-response takes hyperparameters and outputs weights, so approximate it with a hypernet.

Theorem. *Sufficiently powerful hypernets can learn continuous best-response functions, which minimizes the expected loss for any hyperparameter distribution.*

There exists ϕ^*, such that for all $\lambda \in \operatorname{support}(p(\lambda))$,

$$\underset{\text{Train}}{\mathcal{L}} (\mathrm{w}_{\phi^*}(\lambda), \lambda) = \min_{\mathrm{w}} \underset{\text{Train}}{\mathcal{L}} (\mathrm{w}, \lambda)$$

and $\phi^* = \underset{\phi}{\operatorname{argmin}} \underset{p(\lambda')}{\mathbb{E}} \left[\underset{\text{Train}}{\mathcal{L}} (\mathrm{w}_\phi(\lambda'), \lambda') \right]$

Globally Optimizing the Hypernet

- We can learn the best-response without viewing pairs of hyperparameters and optimized weights, by substituting the hypernet output into the training loss. The algorithm is denoted Hyper Training.

initialize ϕ
initialize $\hat{\lambda}$
for T_{hypernet} steps **do**
  $\mathbf{x} \sim$ Training data, $\lambda \sim p(\lambda)$
  $\phi = \phi - \alpha \nabla_\phi \mathcal{L}_{\text{Train}}(\mathbf{x}, \mathrm{w}_\phi(\lambda), \lambda)$
for $T_{\text{hyperparameter}}$ steps **do**
  $\mathbf{x} \sim$ Validation data
  $\hat{\lambda} = \hat{\lambda} - \beta \nabla_{\hat{\lambda}} \mathcal{L}_{\text{Valid.}}(\mathbf{x}, \mathrm{w}_\phi(\hat{\lambda}))$
return $\hat{\lambda}, \mathrm{w}_\phi(\hat{\lambda})$

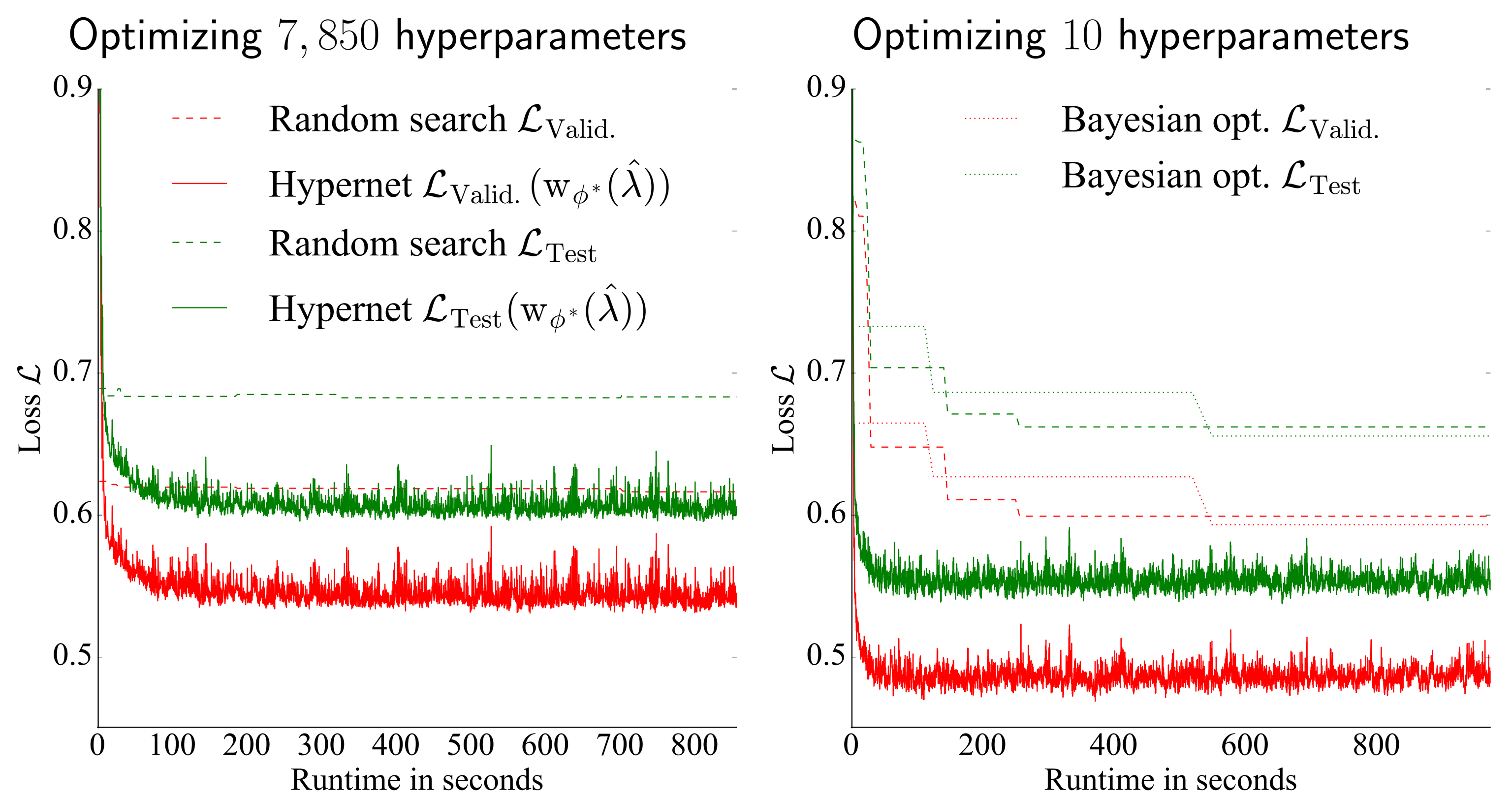
Locally Optimizing the Hypernet

- It is difficult to learn the best-response globally due to finite network size and training time.
- It is easier to learn the best-response locally, update the hyperparameters and repeat.

initialize $\phi, \hat{\lambda}$
for T_{joint} steps **do**
  $\mathbf{x} \sim$ Training data, $\lambda \sim p(\lambda|\hat{\lambda})$
  $\phi = \phi - \alpha \nabla_\phi \mathcal{L}_{\text{Train}}(\mathbf{x}, \mathrm{w}_\phi(\lambda), \lambda)$
  $\mathbf{x} \sim$ Validation data
  $\hat{\lambda} = \hat{\lambda} - \beta \nabla_{\hat{\lambda}} \mathcal{L}_{\text{Valid.}}(\mathbf{x}, \mathrm{w}_\phi(\hat{\lambda}))$
return $\hat{\lambda}, \mathrm{w}_\phi(\hat{\lambda})$

Optimizing 7,850 Hyperparameters

- We investigate our methods performance on tuning hyperparameters of dimensionality 10 and $7,850$.



Benefits of Hyper Training

- Our method provides two potential benefits. These are a better inductive bias by learning the weights instead of loss, and viewing many hyperparameter settings during training.
- We analyze this by comparing our algorithm to Bayesian optimization with 25 samples and a hypernet trained on the same 25 samples.

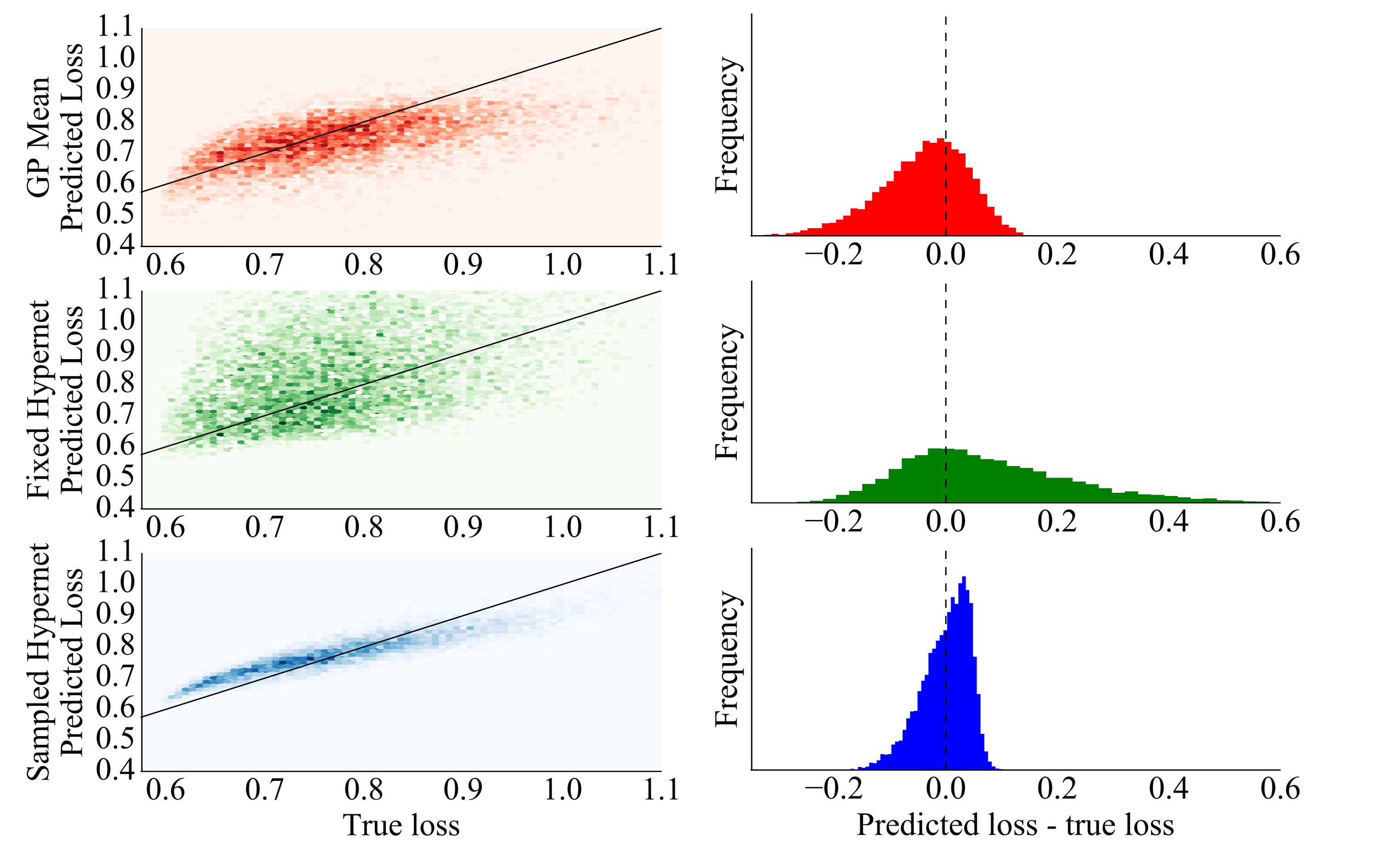


Figure 3: Comparing three approaches to predicting validation loss. *First row:* A Gaussian process, fit on a small set of hyperparameters and the corresponding validation losses. *Second row:* A hypernet, fit on the same small set of hyperparameters and the corresponding optimized weights. *Third row:* Our proposed method, a hypernet trained with stochastically sampled hyperparameters. *Left:* The distribution of predicted and true losses. The diagonal black line is where predicted loss equals true loss. *Right:* The distribution of differences between predicted and true losses. The Gaussian process often under-predicts the true loss, while the hypernet trained on the same data tends to over-predict the true loss.

Conclusions

- We presented an algorithm that efficiently learns a differentiable approximation to a best-response for hyperparameter optimization.
- Hypernets can provide a better inductive bias for hyperparameter optimization than Bayesian optimization.

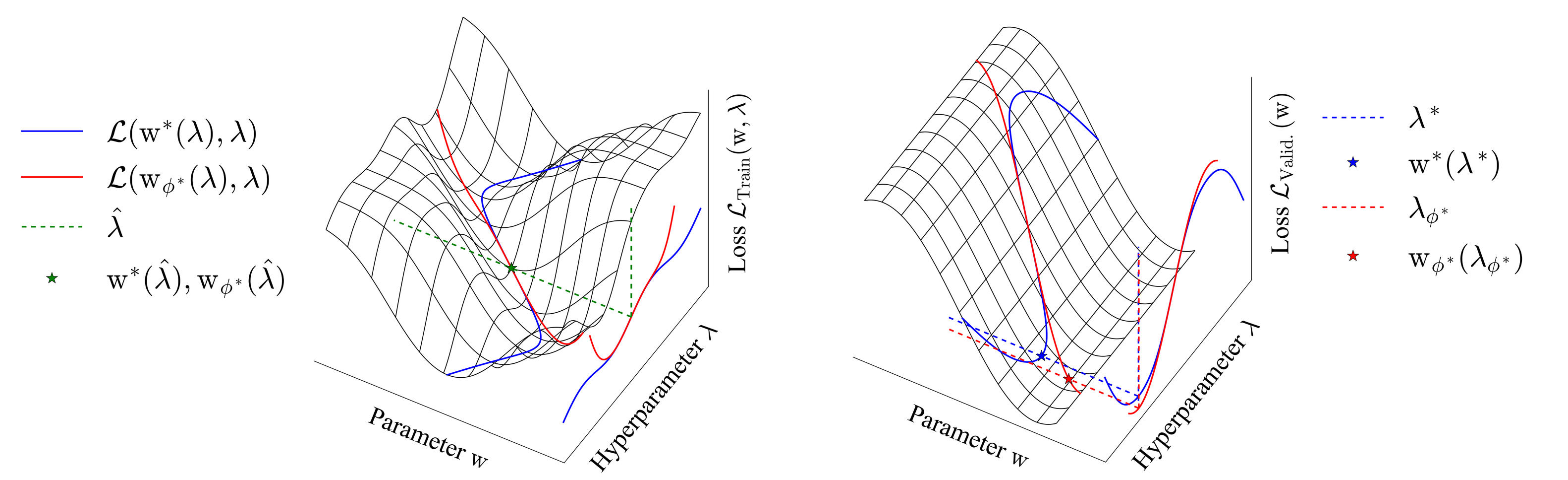


Figure 4: A visualization of exact (blue) and approximate (red) optimal weights as a function of given hyperparameters. *Left:* The training loss surface. *Right:* The validation loss surface. The approximately optimal weights w_{ϕ^*} are output by a linear model fit at $\hat{\lambda}$. The true optimal hyperparameter is λ^*, while the hyperparameter estimated using approximately optimal weights is nearby at λ_{ϕ^*}.