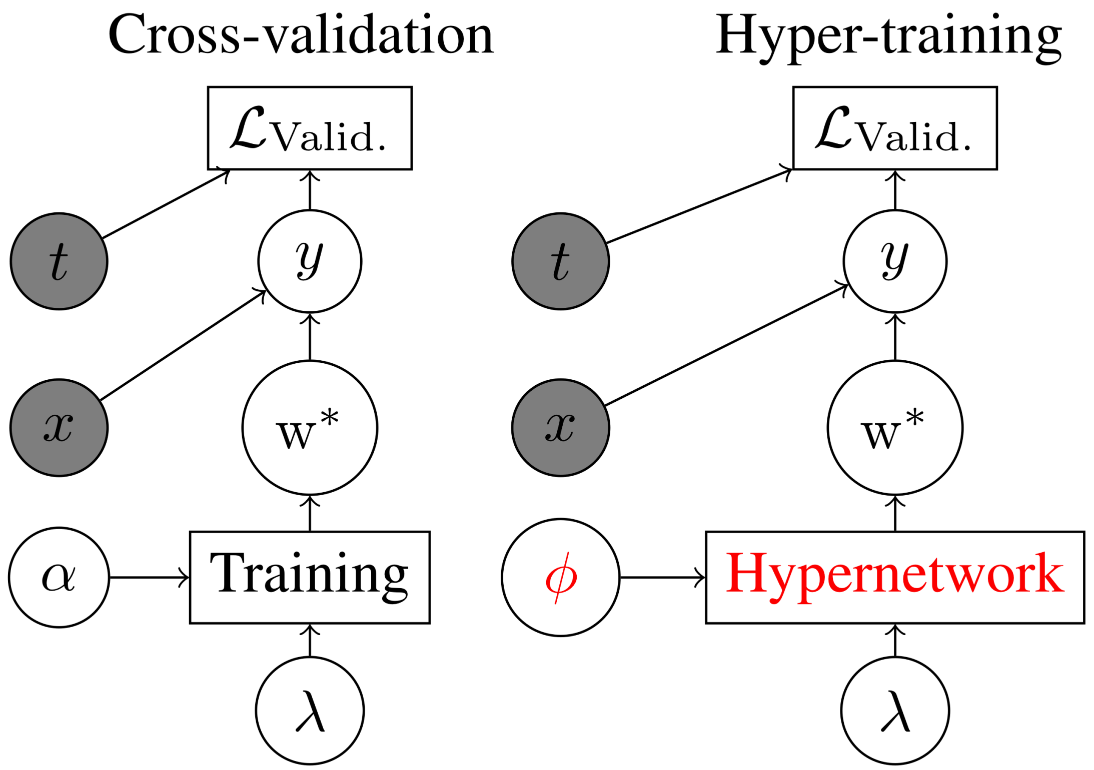
Stochastic Hyperparameter Optimization through Hypernetworks

JONATHAN LORRAINE

& DAVID DUVENAUD

UNIVERSITY OF TORONTO



Cross-validation as Nested Optimization

- Cross-validation nests optimization of network weights inside of optimization of hyperparameters.

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

- Bi-level optimization is a game with a leading player and a following player. Each has their own objective.

- The followers best-responding strategy depends on the leaders strateg

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

- Maclaurin et. Al. (2015) backprop through a training procedure to get gradients, but requires training from scratch each time.

Learning the best-response function

- Lets learn the best-response function and amortize optimization!

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

- New gradient terms:

$$\frac{\partial \mathcal{L}_{\text{Train}}(w_\phi)}{\partial w_\phi} \frac{\partial w_\phi}{\partial \phi} \text{ or } \frac{\partial \mathcal{L}_{\text{Valid.}}(w_\phi(\lambda))}{\partial w_\phi(\lambda)} \frac{\partial w_\phi(\lambda)}{\partial \lambda}$$

Algorithm 1 Standard cross-validation with stochastic optimization

for $i = 1, \dots, T_{\text{outer}}$ **do**
    initialize w
    $\lambda = \text{hyperopt}(\lambda^{(1:i)}, \mathcal{L}_{\text{Valid.}}(w^{(1:i)}))$
    loop
        $\mathbf{x} \sim$ Training data
        $w \mathrel{-}= \alpha \nabla_w \mathcal{L}_{\text{Train}}(w, \lambda, \mathbf{x})$
    $\lambda^i, w^i = \lambda, w$

$i = \underset{i}{\operatorname{argmin}} \mathcal{L}_{\text{Train}}(w^{(i)}, \lambda^{(i)}, \mathbf{x})$

Return $\lambda^{(i)}, w^{(i)}$

Algorithm 2 Optimization of hypernetwork, then hyperparameters

initialize ϕ
initialize $\hat{\lambda}$
loop
    $\mathbf{x} \sim$ Training data, $\lambda \sim p(\lambda)$
    $\phi \mathrel{-}= \alpha \nabla_\phi \mathcal{L}_{\text{Train}}(w_\phi(\lambda), \lambda, \mathbf{x})$

loop
    $\mathbf{x} \sim$ Validation data
    $\hat{\lambda} \mathrel{-}= \beta \nabla_{\hat{\lambda}} \mathcal{L}_{\text{Valid.}}(w_\phi(\hat{\lambda}), \mathbf{x})$

Return $\hat{\lambda}, w_\phi(\hat{\lambda})$

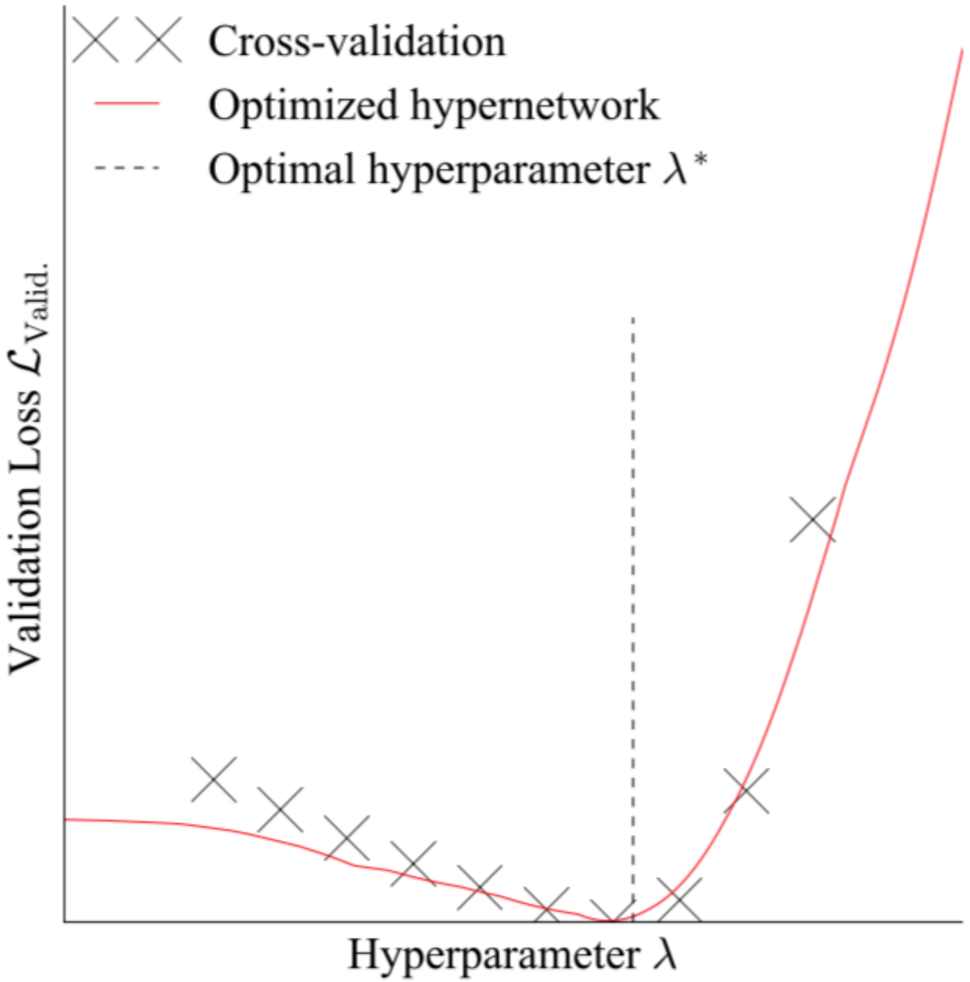


Figure 2: The validation loss of a neural net, estimated by cross-validation (crosses) or by a hypernetwork (line), which outputs $7,850$-dimensional network weights. Cross-validation requires optimizing from scratch each time. The hypernetwork can be used to evaluate the validation loss cheaply.

Local Optimization

- Limited capacity hypernetwork in practice.

Algorithm 2 Optimization of hypernetwork, then hyperparameters

initialize ϕ
initialize $\hat{\lambda}$
loop
  $\mathbf{x} \sim$ Training data, $\lambda \sim p(\lambda)$
  $\phi -= \alpha \nabla_\phi \mathcal{L}_{\text{Train}}(\mathrm{w}_\phi(\lambda), \lambda, \mathbf{x})$

loop
  $\mathbf{x} \sim$ Validation data
  $\hat{\lambda} -= \beta \nabla_{\hat{\lambda}} \mathcal{L}_{\text{Valid.}}(\mathrm{w}_\phi(\hat{\lambda}), \mathbf{x})$
Return $\hat{\lambda}, \mathrm{w}_\phi(\hat{\lambda})$

Algorithm 3 Joint optimization of hypernetwork and hyperparameters

initialize ϕ
initialize $\hat{\lambda}$
loop
  $\mathbf{x} \sim$ Training data, $\lambda \sim p(\lambda|\hat{\lambda})$
  $\phi -= \alpha \nabla_\phi \mathcal{L}_{\text{Train}}(\mathrm{w}_\phi(\lambda), \lambda, \mathbf{x})$

  $\mathbf{x} \sim$ Validation data
  $\hat{\lambda} -= \beta \nabla_{\hat{\lambda}} \mathcal{L}_{\text{Valid.}}(\mathrm{w}_\phi(\hat{\lambda}), \mathbf{x})$
Return $\hat{\lambda}, \mathrm{w}_\phi(\hat{\lambda})$

Algorithm 4 Simplified joint optimization of hypernetwork and hyperparameters

initialize $\phi, \hat{\lambda}$
loop
  $\mathbf{x} \sim$ Training data, $\mathbf{x}' \sim$ Validation data
  $\phi -= \alpha \nabla_\phi \mathcal{L}_{\text{Train}}(\mathrm{w}_\phi(\hat{\lambda}), \hat{\lambda}, \mathbf{x})$
  $\hat{\lambda} -= \beta \nabla_{\hat{\lambda}} \mathcal{L}_{\text{Valid.}}(\mathrm{w}_\phi(\hat{\lambda}), \mathbf{x}')$
Return $\hat{\lambda}, \mathrm{w}_\phi(\hat{\lambda})$

- Learn the best-response in some small neighborhood about our current hyperparameter.

Global:

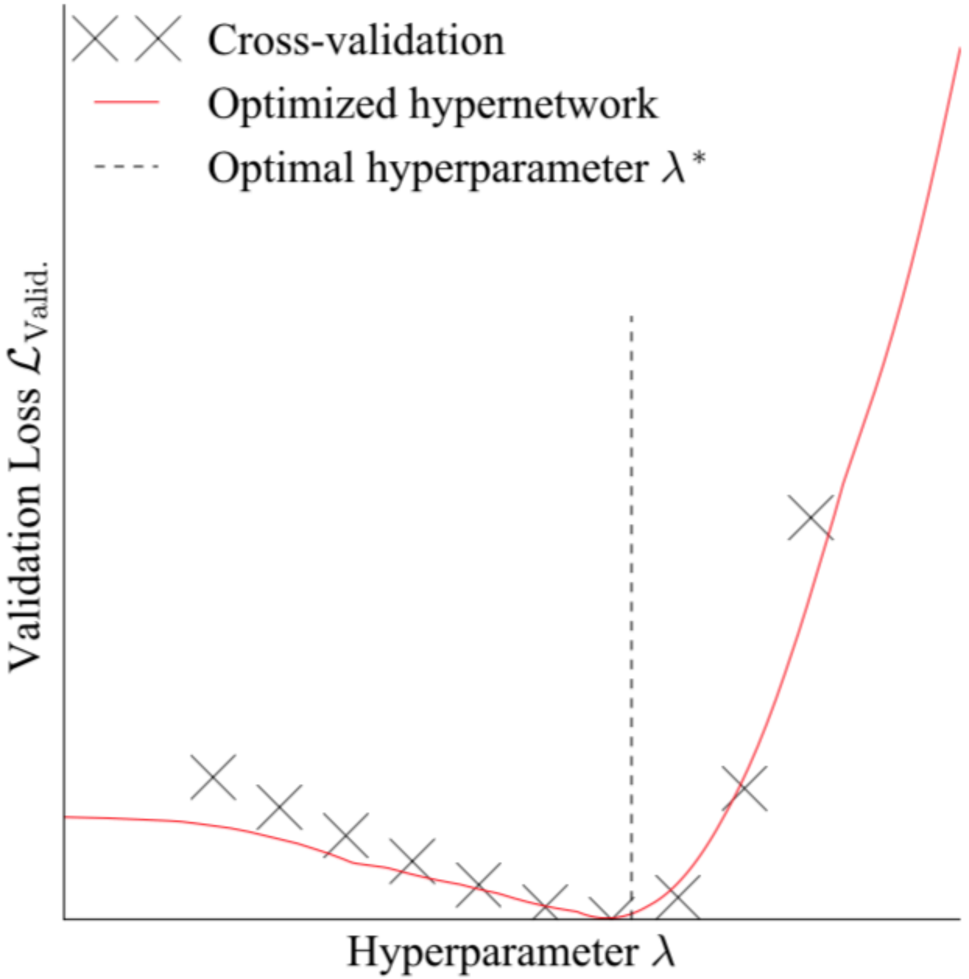


Figure 2: The validation loss of a neural net, estimated by cross-validation (crosses) or by a hypernetwork (line), which outputs 7, 850-dimensional network weights. Cross-validation requires optimizing from scratch each time. The hypernetwork can be used to evaluate the validation loss cheaply.

Local:

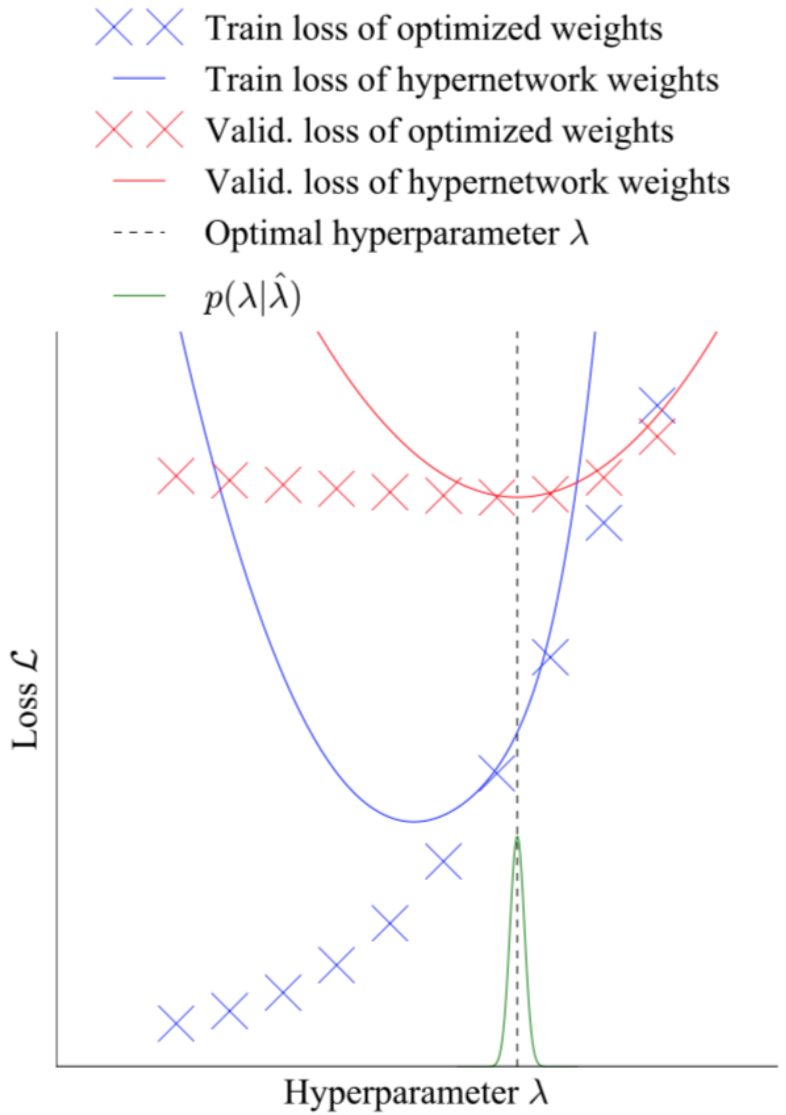


Figure 4: Training and validation losses of a neural network, estimated by cross-validation (crosses) or a linear hypernetwork (lines). The hypernetwork's limited capacity makes it only accurate where the hyperparameter distribution puts mass.

Visualization

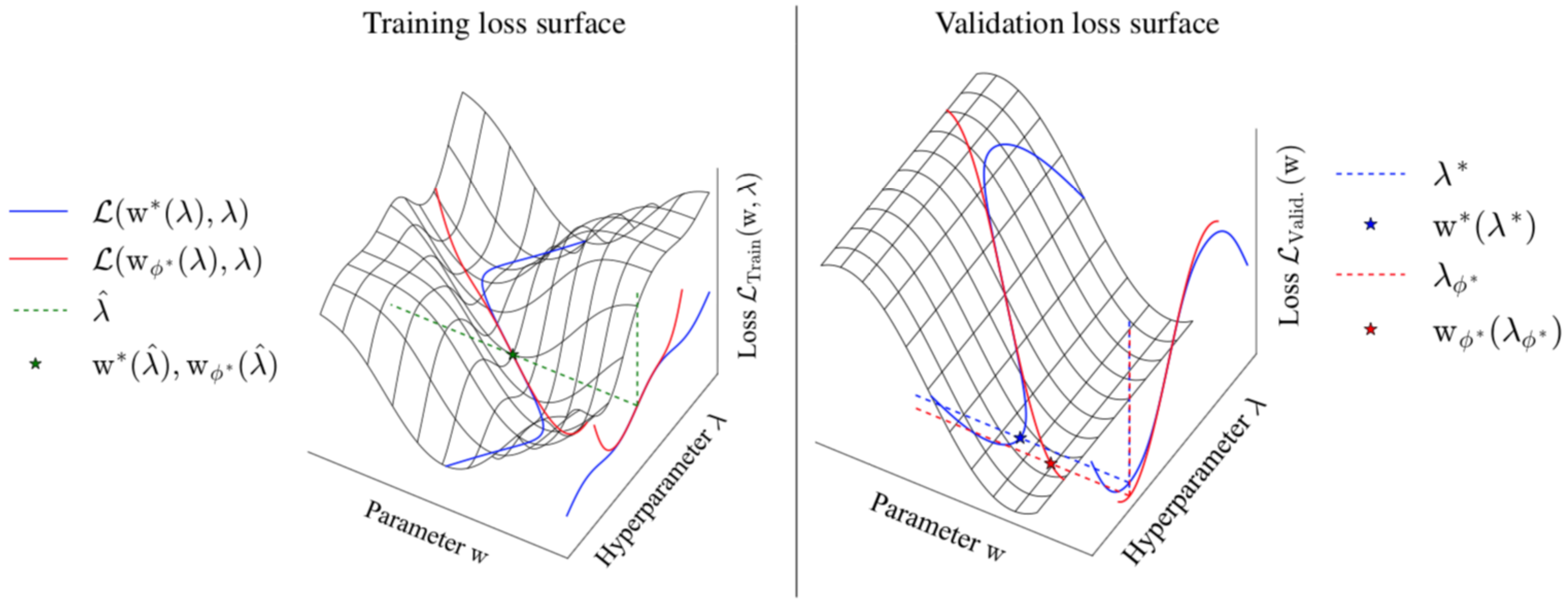
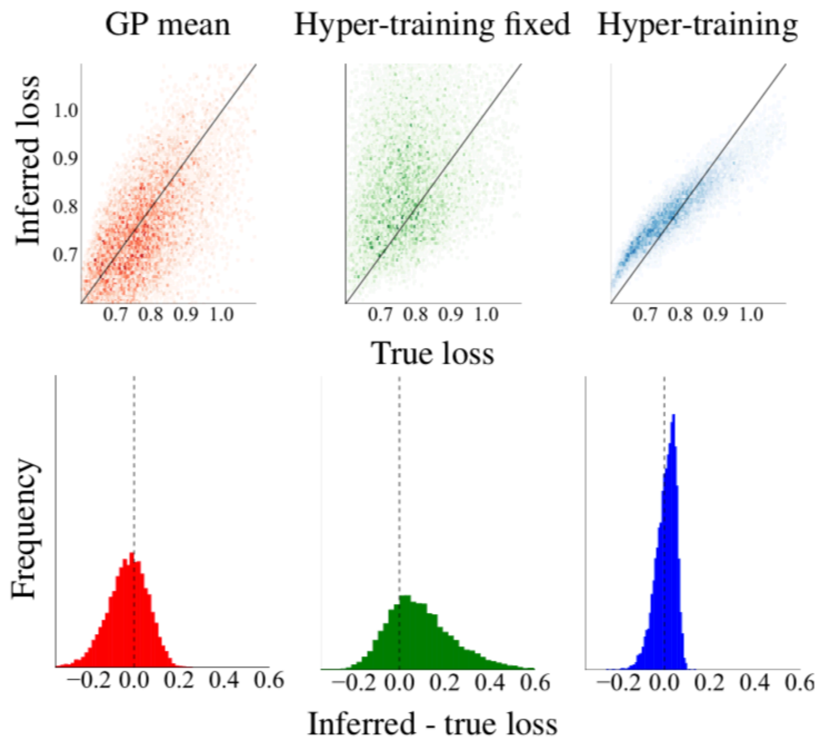


Figure 3: A visualization of exact (blue) and approximate (red) optimal weights as a function of hyperparameters. The approximately optimal weights w_{ϕ^*} are output by a linear model fit at $\hat{\lambda}$. The true optimal hyperparameter is λ^*, while the hyperparameter estimated using approximately optimal weights is nearby at λ_{ϕ^*}.

Stochastic evaluation of validation loss

Idea: Hyper-training is effective because it partially optimizes across many hyperparameters.



Limitations

- No inner optimization parameters can be tuned (they don't exist!).
- Hard to tune discrete hyperparameters with gradients (working on it).
- No uncertainty based exploration.
- Hard to choose the distribution of hyperparameters to train against.
  - Future work: use implicit function theorem instead?

Takeaway and Future Directions

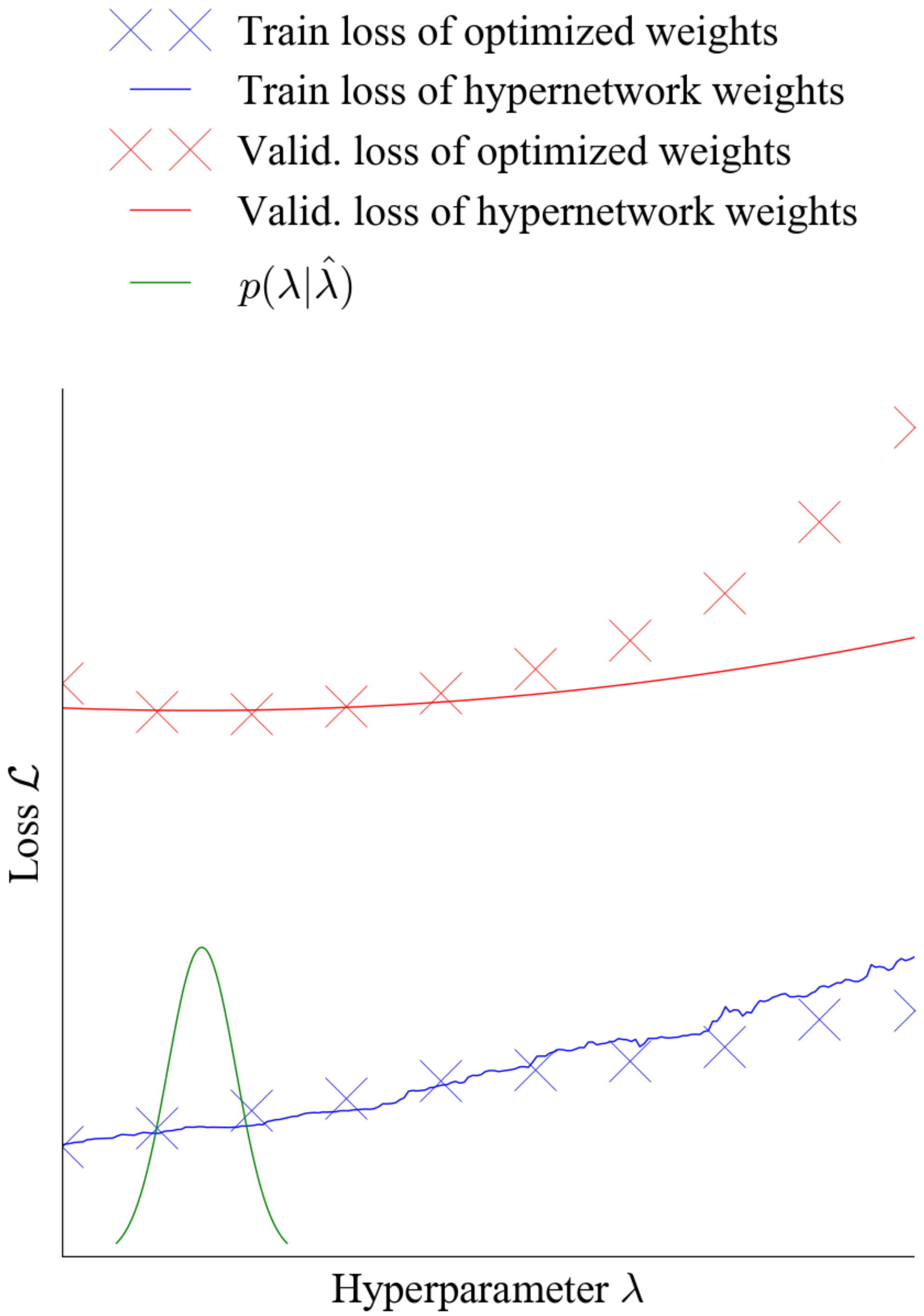
- Currently using a linear hypernet - consider a net with 10,000,000 weights and 10 hyperparameters.

- Try optimizing other hyperparameters: e.g. training data

$$\mathrm{w}^*(\lambda) = \underset{\mathrm{w}}{\mathrm{argmin}} \sum_{\mathrm{x}_i, \mathrm{t}_i \in \mathcal{D}_t} \mathcal{L}_{\mathrm{p}}\left(\mathrm{y}_{\mathrm{w}}\left(\mathrm{x}_i\right), \mathrm{t}_i\right) + \mathcal{L}_{\mathrm{r}}\left(\mathrm{y}_{\mathrm{w}}, \mathrm{X}_t\right)$$

- Main point: Learning the best-response function lets you collapse a nested optimization problem into a joint optimization problem. Can be applied to GANs, and possibly to funding Nash equilibria more generally.

Optimizing weight dropout on MNIST.



Related Work

- Brock, Andrew, Lim, Theodore, Ritchie, JM, and Weston, Nick. SMASH: One-shot model architecture search through hypernetworks. *arXiv:1708.05344*, 2017.
- MAML
- Efficient Neural Architecture search