A. Parameter Sweep Strategy

To determine the best-performing hyper-parameter setting, we evaluate each configuration using 10 independent runs initialized with a different random seed, leading to as many learning curves. We averaged the learning curves and summed the second half of the averaged learning curves to obtain a single number representing the performance of the particular configuration. If the best-performing parameter setting falls on the boundary of the range of tested values for any hyper-parameter, we widened the range until this was not true. We evaluated the best-performing configuration using 30 additional runs, each initialized with a different random seed. We report the average learning curve, along with its standard error. In all experiments, the resolution of the reported curves is 2,000 steps: we log the average returns of the most recent 20 episodes every 2,000 steps.

B. Baseline Algorithms

B.1. Deep Q-Networks (DQN)

We estimate DQN's action-value function using a fully connected neural network with ReLU activations. We repeat all experiments with four different fully connected neural network architectures: 1 hidden layer with 64 hidden units, 1 hidden layer with 128 hidden units, 2 hidden layers with 64 hidden units each, and 2 hidden layers with 128 hidden units each. For each network architecture, we determine the best setting for the step-size, the batch size, and the replay memory size by sweeping over possible parameter configurations. For DQN baseline, the range of values for the parameter sweep, and the configuration of the rest of the hyper-parameters are presented in Table 1.

B.2. Model-Based Value Expansion (MVE)

We implement MVE by extending the DQN algorithm with the model-based policy evaluation technique described in Figure 1 of the main paper. We instantiate MVE with a deterministic model learned using the squared error loss. We assume the reward signal to be known; that is, we only learn the dynamics function. We study the effect of model capacity by progressively reducing the size of the neural network used for model learning. In particular, we use four variants of a single hidden layer neural network, which vary only in the number of hidden units: 128 hidden units, 64 hidden units, 16 hidden units, and 4 hidden units. In all cases, we learn the model online using the experience gathered in the replay buffer: at every time-step, alongside the MVE value function update, we separately sample a batch of transitions to update the model.

Once we have identified the best hyper-parameter configuration for the DQN baselines which vary in their value function architecture, we keep the same hyper-parameter configuration for their MVE extensions, and only sweep over the model learning rate for each of the four model architectures. The range of values for the parameter sweep, and the configuration of the rest of the hyper-parameters are presented in Table 2.

C. Selective Model-based Value Expansion: Additional Details and Results

Learned Variance Implementation Details: We modify the base neural networks for the dynamics function to output the diagonal covariances alongside the mean next-state vector. We enforce the positivity constraint on the covariances by passing the corresponding output through the *softplus* function $\log(1 + \exp(\cdot))$; and, for numerical stability, we also add a small constant value of 10^{-6} to the predicted covariances (Lakshminarayanan et al., 2017). The model is optimized using the loss function

$$L_{(s,a,s)}(\theta) = [\mu_\theta(s,a) - s']^T \Sigma_\theta^{-1}(s,a)[\mu_\theta(s,a) - s'] \\ + \log\det\Sigma_\theta(s,a),$$

where $\Sigma_\theta(s,a)$ is assumed to be diagonal. For input (s_i, a_i), $\sigma^2(s_i, a_i)$ is the trace of $\Sigma_\theta(s_i, a_i)$. The range of values for the parameter sweep, and the configuration of the rest of the hyperparameters are presented in Table 3.

Additional Results for Selective MVE with Learned Variance. We present Acrobot results for the value function network architectures with a single hidden layer and 64 hidden units (Figure 10-12), with 2 hidden layers and 64 hidden units each (Figure 13-15), and with a 2 hidden layer with 128 hidden units each (Figure 16-18). All reported curves are obtained by averaging 30 runs; the shaded regions represent the standard error. These results are consistent with the discussion in the main paper and provide additional evidence for the utility of learned variance for selective planning. For instance, they suggest that selective planning is useful even when the value function itself has restricted capacity—the network with only 64 hidden units, for example.

Selective MVE with ensemble variance. The ensemble-based selective MVE is exactly like the learned-variance variant except for one difference: the uncertainty, $\sigma(s,a)$, is the variance of the predictions made by the individual members of the ensemble. (We add the components of the variance vector to obtain a single number.) The range of values for the parameter sweep, and the configuration of the rest of the hyperparameters are presented in Table 4.

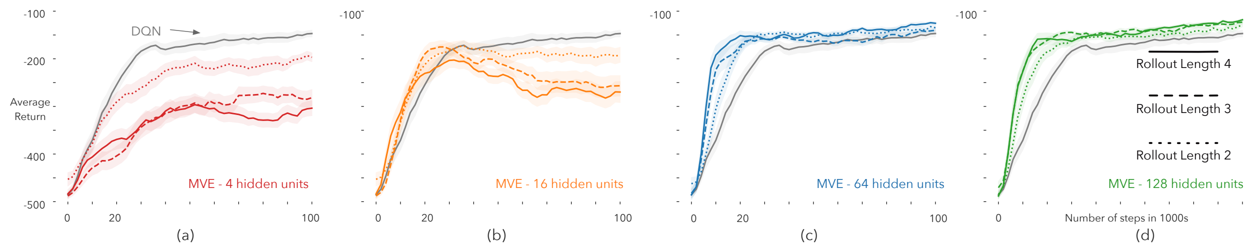

Figure 10. The effect of model capacity on MVE's performance with the value function network consisting of a single hidden layer and 64 hidden units.

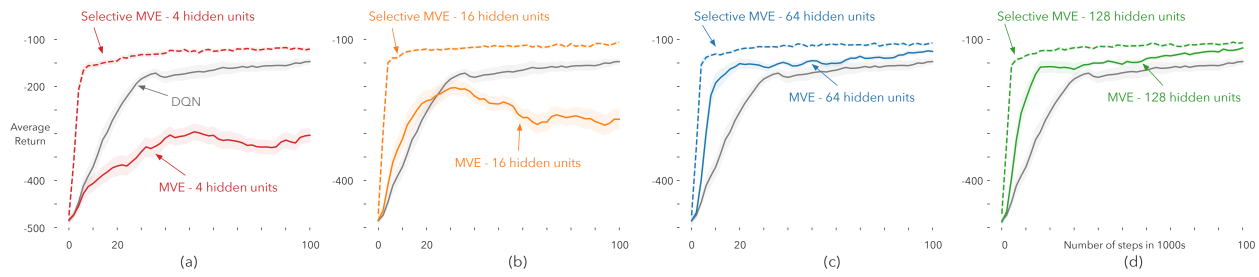

Figure 11. Results of selective MVE ($\tau = 0.1$) for the value function network consisting of a single hidden layer and 64 hidden units.

D. Regression Example: Additional Details and Results

In Section 4's regression experiment, we use Adam optimizer (Kingma & Ba, 2015) for all the methods. We set the batch size to 16. We consider the learning rates 0.01, 0.001, and 0.0001. We use ReLU activations for non-linearities, and initialize the networks with Glorot initialization (Glorot & Bengio, 2010).

For Monte Carlo dropout, we set the dropout probability $p = 0.1$. To obtain the variance, we perform 10 stochastic forward passes.

For the ensemble method, we use an ensemble of 10 neural networks. All networks in the ensemble are trained using the squared-error loss.

For randomized prior functions with bootstrapping, we train each member of the ensemble on a bootstrapped dataset generated from the original dataset by randomly sampling with replacement.

For heteroscedastic regression, we train separate neural networks for the mean and the variance, and optimize them jointly using the loss from Equation 1 (main paper). While we change the capacity of the mean network across the three regimes (large network, medium-sized network, and small network), we restrict the variance network to be small—a single hidden layer with 64 hidden units—in all three regimes.

For each uncertainty method, every configuration is evaluated using 5 independent runs initialized with a different random seed. While the results remain consistent across the independent runs, we present results for a single run chosen randomly. We present results for additional configurations of the learning rate in Figure 19-20.

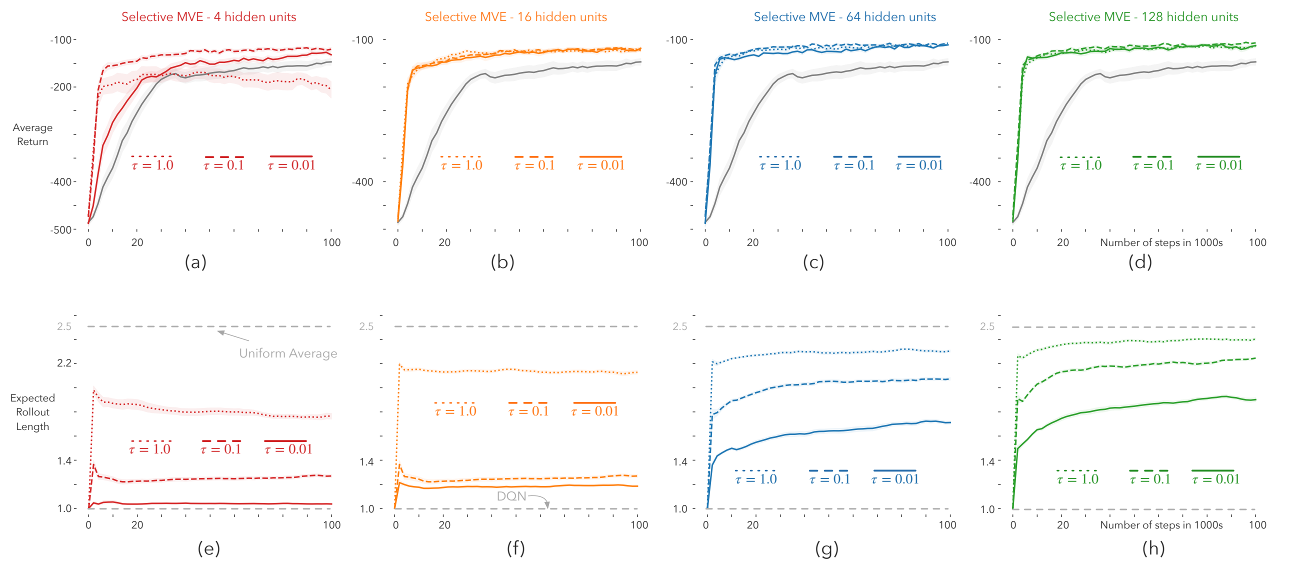

Figure 12. Effect of τ on the performance of Selective MVE with the value function network consisting of a single hidden layer and 64 hidden units.

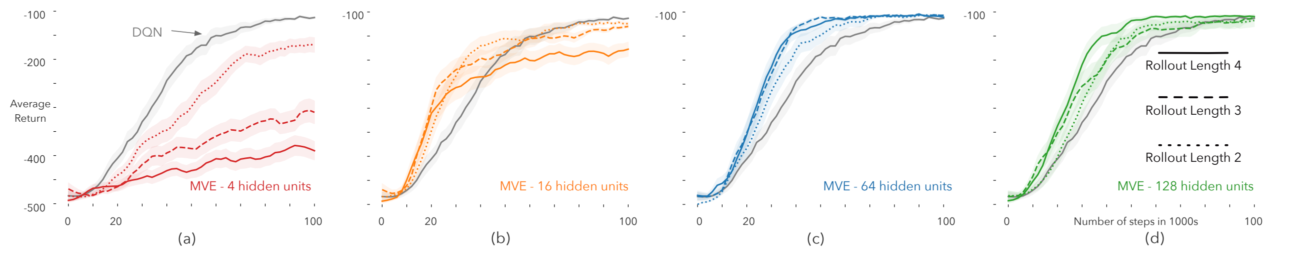

Figure 13. The effect of model capacity on MVE's performance with the value function network consisting of 2 hidden layers with 64 hidden units each.

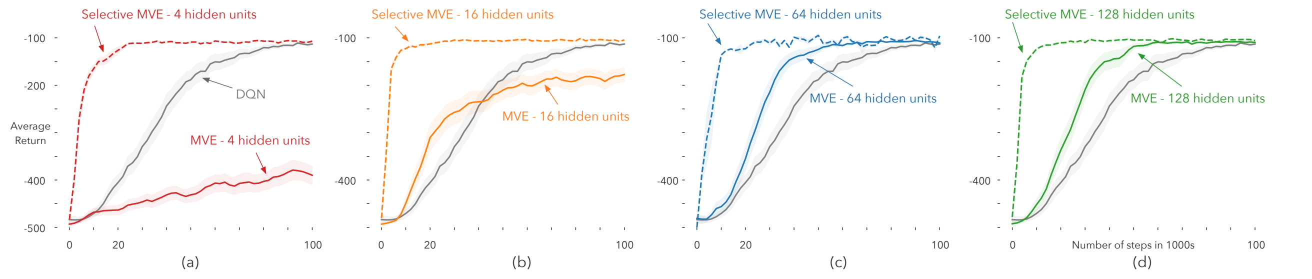

Figure 14. Results of selective MVE ($\tau = 0.1$) with value function network of consisting of 2 hidden layers with 64 hidden units each.

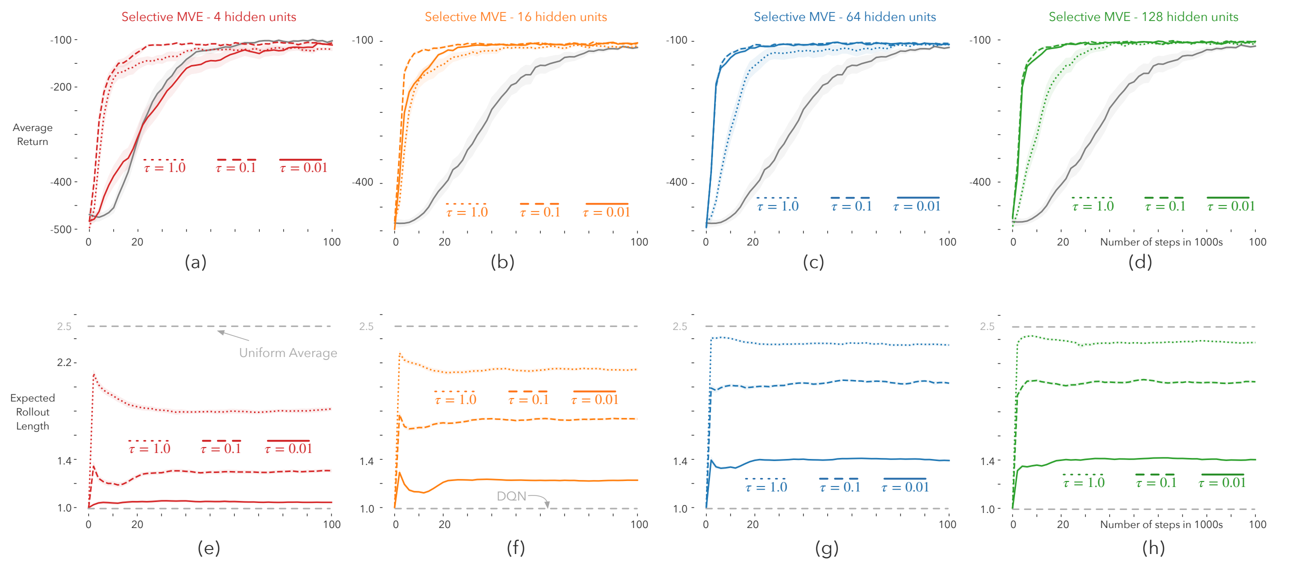

Figure 15. Effect of τ on the performance of Selective MVE with the value function network consisting of hidden layers with 64 hidden units each.

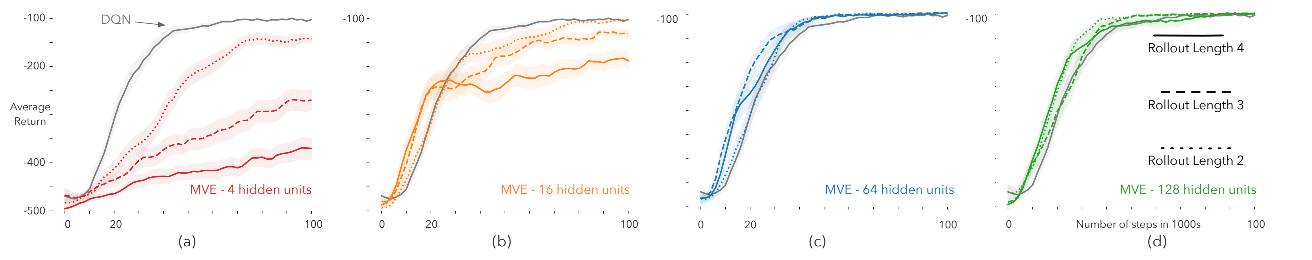

Figure 16. The effect of model capacity on MVE's performance with the value function network consisting of 2 hidden layers with 128 hidden units each.

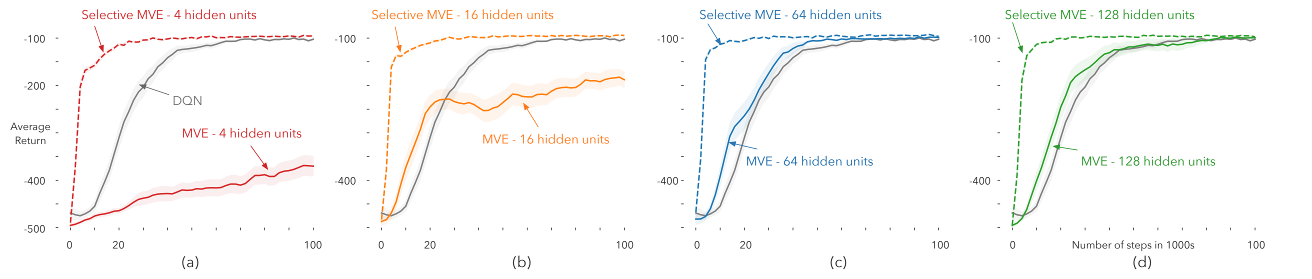

Figure 17. Results of selective MVE ($\tau = 0.1$) with the value function network consisting 2 hidden layers with 128 hidden units each.

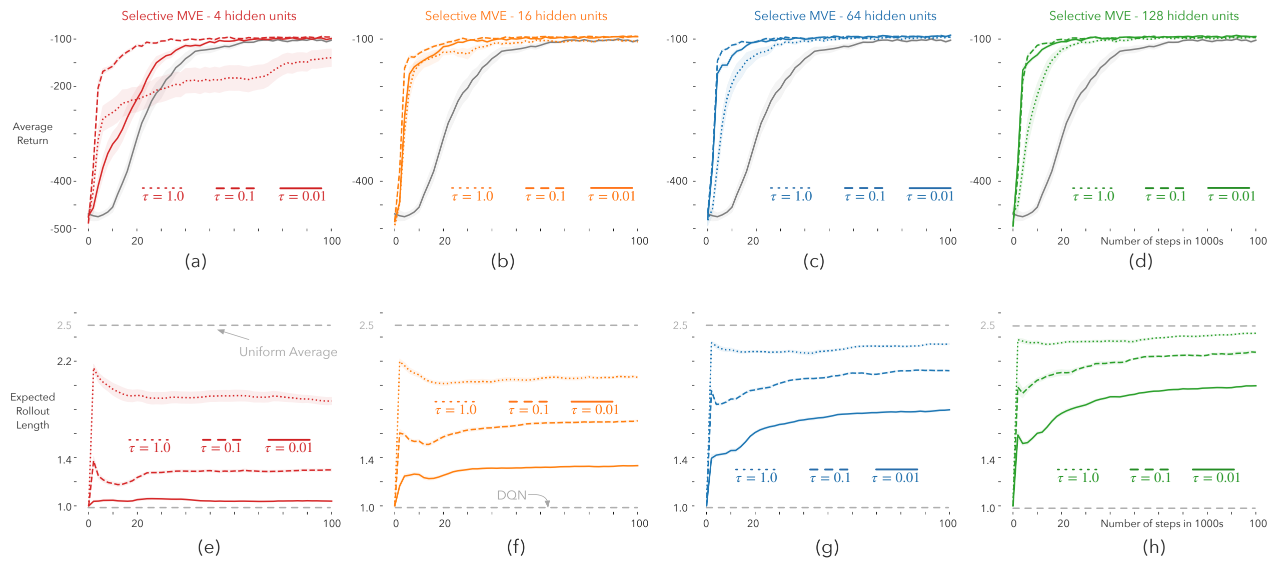

Figure 18. Effect of τ on the performance of Selective MVE with the value function network consisting of 2 hidden layers with 128 hidden units each.

Table 1. DQN hyperparameters used in the experiments. The step-size, the batch size, and the replay memory size were determined by sweeping over the range specified in the respective rows.

Hyperparameter	**Values**
Optimizer	RMSProp
Step-size (α)	0.03, 0.01, 0.003, 0.001, 0.0003, 0.0001
Batch size	16, 32, 64
Replay memory size	10000, 20000, 50000
Target network update frequency	256 environment steps
Training frequency	1 update for every environment step
Exploration rate (ϵ)	0.1
Discount factor (γ)	1.0

Table 2. MVE specific hyperparameters. For each simulated trajectory length (rollout length), the model learning step-size (β) was determined by sweeping over the range specified in the respective row.

Hyperparameter	**Values**
Optimizer	Adam
Model learning step-size (β)	0.1, 0.01, 0.001, 0.0001
Batch size	16
Loss function	Homoscedastic (MSE)
Model learning frequency	1 update for every environment step
Simulated trajectory length	2, 3, 4

Table 3. Hyperparameters for Selective-MVE. Model learning step-size (β) was determined by sweeping over the range specified in the respective row.

Hyperparameter	**Values**
Optimizer	Adam
Model learning step-size (β)	0.1, 0.01, 0.001, 0.0001
Batch size	16
Loss function	Heteroscedastic
Model learning frequency	1 update for every environment step
Simulated trajectory length	4
Softmax temperature (τ)	0.1

Table 4. Hyperparameters for ensemble-based Selective MVE in Acrobot. Model learning step-size (β) was determined by sweeping over the range specified in the respective row.

Hyperparameter	**Values**
Optimizer	Adam
Model learning step-size (β)	0.1, 0.01, 0.001, 0.0001
Batch size	16
Loss function	Homoscedastic (MSE)
Model learning frequency	1 update for every environment step
Simulated trajectory length	4
Softmax temperature (τ)	0.1
Number of networks	5

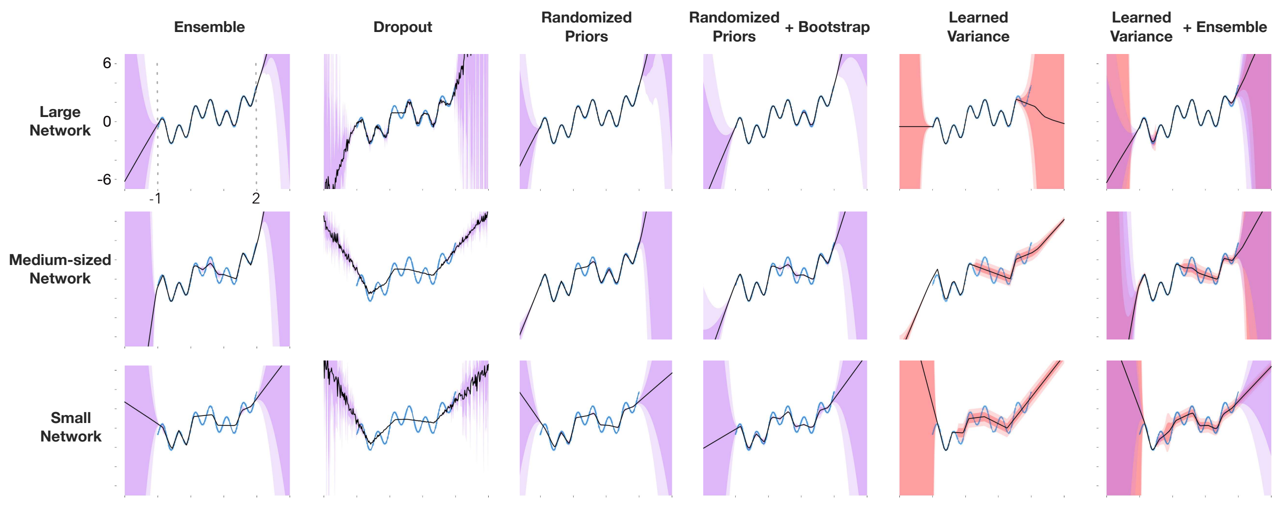

Figure 19. Regression results for the learning rate 0.01.

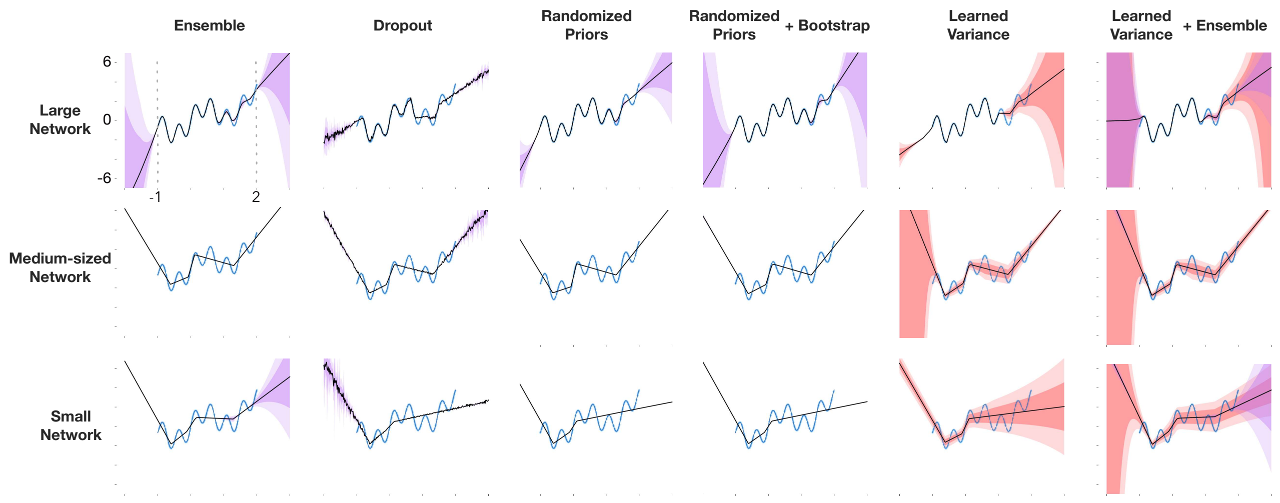

Figure 20. Regression results for the learning rate 0.0001.