6th ICML Workshop on Automated Machine Learning (2019)

Probabilistic Rollouts for Learning Curve Extrapolation Across Hyperparameter Settings

M. Gargiani GARGIANI@INFORMATIK.UNI-FREIBURG.DE
University of Freiburg, Germany

A. Klein KLEINAA@INFORMATIK.UNI-FREIBURG.DE
University of Freiburg, Germany

S. Falkner STEFAN.FALKNER@DE.BOSCH.COM
Bosch Center for Artificial Intelligence, Germany

F. Hutter FH@CS.UNI-FREIBURG.DE
University of Freiburg, Germany
Bosch Center for Artificial Intelligence, Germany

Abstract

We propose probabilistic models that can extrapolate learning curves of iterative machine learning algorithms, such as stochastic gradient descent for training deep networks, based on training data with variable-length learning curves. We study instantiations of this framework based on random forests and Bayesian recurrent neural networks. Our experiments show that these models yield better predictions than state-of-the-art models from the hyperparameter optimization literature when extrapolating the performance of neural networks trained with different hyperparameter settings.

1. Introduction

The efficient optimization of machine learning hyperparameters is one of the most basic yet most important tasks in automated machine learning (AutoML, Hutter et al. (2018)). E.g., hyperparameter optimization has already achieved remarkable improvements of the state-of-the-art in different applications, such as natural language processing (Melis et al., 2018) or AlphaGO (Chen et al., 2018). A wide range of hyperparameter optimization methods exists (see, e.g., Feurer and Hutter (2018) for an overview), and since the objective function of interest (e.g., cross-validation error) is typically expensive, the most efficient methods tend to leverage cheap-to-evaluate proxies (so-called fidelities) (Swersky et al., 2014; Domhan et al., 2015; Baker et al., 2017; Kandasamy et al., 2017; Klein et al., 2017a,b; Li et al., 2017; Falkner et al., 2018).

A frequently used fidelity for iterative machine learning algorithms is the performance over time or iterations, the so-called *learning curve*: the early performance of a network architecture or hyperparameter configuration is typically quite indicative of its final performance when trained to convergence. Some approaches model these learning curves to decide whether to stop or continue the evaluation of a hyperparameter configuration (Swersky et al., 2014; Domhan et al., 2015; Baker et al., 2017; Klein et al., 2017b; Li et al., 2017; Falkner et al., 2018), while others actively choose a budget before evaluating in order to maximize the information gained per time spent (Klein et al., 2017a; Kandasamy et al., 2017).

Another key difference between previous methods lies in what the model predicts based on what information. Several approaches (Swersky et al., 2014; Kandasamy et al., 2017; Klein et al., 2017b,a) build a global model capable of predicting the performance at any fidelity based on the hyperparameter configuration alone. Others (Baker et al., 2017; Falkner et al., 2018) only train models that predict the learning curve for a fixed set of fidelities, and a third group (Li et al., 2017; Domhan et al., 2015) only operates on single learning curves and extrapolates them without taking the hyperparameter configuration into account.

A final notable distinction are the assumptions going into the model. Many existing methods use hand-designed basis functions describing common characteristics of learning curves (Domhan et al., 2015; Klein et al., 2017a,b; Swersky et al., 2014), while others (Baker et al., 2017; Kandasamy et al., 2017; Li et al., 2017; Falkner et al., 2018) make no or very weak assumptions about the shape of the learning curves, but rely more heavily on observed training data.

Surprisingly, none of the existing methods truly takes into account the sequential nature of learning curves by using a sequence model that can be rolled out for an arbitrary number of time steps. In this paper, we fill this gap; our contributions are as follows:

- We introduce the first sequence models for learning curve prediction. We provide instantiations based on random forests and Bayesian neural networks that also take hyperparameter configurations into account.
- These sequence models are the first that can cheaply generate extrapolations of partially observed learning curves with similar characteristics to those in the training data.
- In preliminary experiments, we show that these models are not only more flexible and accurate than previous learning curve models, but also allow to efficiently transfer knowledge to new tasks with the same input domain.

2. Probabilistic Prediction of Learning Curves

Previous work (Swersky et al., 2014; Klein et al., 2017b) casts the prediction $\tilde{y}_t \in \mathbb{R}$ of the performance $y_t \in \mathbb{R}$ of a hyperparameter configuration $\boldsymbol{\theta} \in \Theta$ at a time step $t \in \mathbb{R}$ as a mapping $\tilde{y}_t = g(\mathbf{x}_t; \boldsymbol{\omega})$ with $\mathbf{x}_t = [\boldsymbol{\theta}^\top, t]^\top$ and $\boldsymbol{\omega}$ being the collection of the model parameters.

Instead, we treat learning curves as sequential time series and predict the value at the current time step based on the values observed at previous time steps. More formally, we keep the same mapping $\tilde{y}_t = g(\mathbf{x}_t; \boldsymbol{\omega})$ but augment the input $\mathbf{x}_t = \left[\boldsymbol{\theta}^\top, y_{t-K-1}, \ldots, y_{t-1}\right]^\top$ by the past K observed points $y_{t-K-1}, \ldots, y_{t-1}$. We assume that the unknown true objective function $f(\boldsymbol{\theta}, t)$ is only observable with noise $y_t = f(\boldsymbol{\theta}, t) + \epsilon$, with $\epsilon \sim \mathcal{N}(0, \sigma^2)$. To predict for an unseen data point $\mathbf{x}_t^\star$ during inference time, we approximate the predictive distribution by a Gaussian:

$$p(y_t^\star \mid \mathbf{x}_t^\star, \mathcal{D}) \approx \mathcal{N}\big(\mu(y_t^\star \mid \mathbf{x}_t^\star, \mathcal{D}), \sigma^2(y_t^\star \mid \mathbf{x}_t^\star, \mathcal{D})\big) \tag{1}$$

where $\mathcal{D} = \{\mathbf{x}^{(0)}, \mathbf{y}^{(0)}, \ldots \mathbf{x}^{(N-1)}, \mathbf{y}^{(N-1)}\}$ is the training dataset that consists of N learning curves with potentially varying lengths, together with their corresponding hyperparameter configuration vectors. We now describe how to predict $\mu(y_t^\star \mid \mathbf{x}_t^\star, \mathcal{D})$ and $\sigma^2(y_t^\star \mid \mathbf{x}_t^\star, \mathcal{D})$ in

Equation 1 using two different probabilistic regression models: random forests (RFs) and variational recurrent neural networks (VRNNs).

2.1 Random Forests

First, we consider random forests (Breimann, 2001) because of their conceptual simplicity and practical robustness against their own hyperparameters. Following Hutter et al. (2014), given a forest with B trees, each tree i stores the empirical mean $\tilde{\mu}_i$ and variance $\tilde{\sigma}_i^2$ and, for a test point $\mathbf{x}_t^\star$, the forest returns a Gaussian predictive distribution $\mathcal{N}(\tilde{\mu}(y_t^\star \mid \mathbf{x}_t^\star, \mathcal{D}), \tilde{\sigma}^2(y_t^\star \mid \mathbf{x}_t^\star, \mathcal{D}))$ where $\tilde{\mu}(y_t^\star \mid \mathbf{x}_t^\star, \mathcal{D}) = 1/B \sum_i \tilde{\mu}_i$ is the mean of the individual tree predictions and $\tilde{\sigma}^2(y_t^\star \mid \mathbf{x}_t^*, \mathcal{D}) = 1/B \cdot \sum_i \tilde{\sigma}_i^2 + 1/B \cdot \sum_i [\tilde{\mu}_i - \tilde{\mu}(y_t^\star \mid \mathbf{x}_t^*, \mathcal{D})]^2$ is computed based on the law of total variance. At inference time, this model requires access to the first K points of an unseen learning curve, but can then extend these to arbitrary length, which we call a *roll out*. For a single roll out, we sample y_{K+1}^* from the predictive distribution defined above. This process can then be consecutively applied until a whole sequence $[\tilde{y}_{K+1}^r, \ldots, \tilde{y}_T^r]$ is generated up to some time step T. By averaging over R independent roll outs, we approximate Equation 1 by a Gaussian with mean $\mu(y_t^\star \mid \mathbf{x}_t^\star, \mathcal{D}) = \frac{1}{R} \sum_{r=1}^R \tilde{y}_t^r$ and variance $\sigma^2(y_t^\star \mid \mathbf{x}_t^\star, \mathcal{D}) = \frac{1}{R} \sum_{r=1}^R (\tilde{y}_t^r - \mu(y_t^\star \mid \mathbf{x}_t^\star, \mathcal{D}))^2$.

2.2 Variational Recurrent Neural Networks (VRNNs)

Due to their success for time series prediction (e.g. Rangapuram et al. (2018)), we also consider recurrent neural networks in form of long short-term memory (LSTM) cells (Hochreiter and Schmidhuber, 1997). To obtain uncertainty estimates, we use variational dropout (Gal and Ghahramani, 2016a,b) to allow for a Bayesian treatment of the weights. Given a hyperparameter configuration $\boldsymbol{\theta}$, a dropout rate $d \in (0, 1)$, and the previous observed point in the learning curve y_{t-1}, we predict the next step by:

$$\tilde{y}_t = h_3 \left(r_2 \left(\mathbf{h}_2 \odot \mathbf{z}_2, r_1 \left(\mathbf{h}_1 \odot \mathbf{z}_1, y_{t-1} \right) \right) \right),$$

where

$$\mathbf{h}_1 = h_1(\boldsymbol{\theta}) \qquad \mathbf{h}_2 = h_2(\boldsymbol{\theta})$$
$$\mathbf{z}_1 \sim Bernoulli(1-d) \qquad \mathbf{z}_2 \sim Bernoulli(1-d)$$

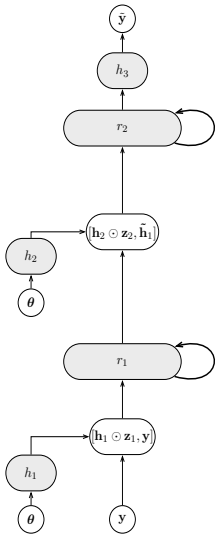


Figure 1: VRNN model.

and $\odot$ denotes the element-wise product, $r_i(\cdot)$ are LSTM blocks and $h_i(\cdot)$ feedforward neural networks. See Figure 1 and Section A in the Appendix for a graphical representation of our model, where we used $\tilde{\mathbf{h}}_1$ to indicate the output of r_1. As described above for the random forest, to predict for an unobserved point in a learning curve $y_t^\star$ at any time step t, we first perform R rollouts by keeping dropout active and feed the prediction of our model back to itself. The final prediction is then the mean and variance of the rollouts (*MC dropout*). Note that, compared to the random forest, we set $K = 1$ and implicitly accumulate memory of the previous observed points in the hidden state of the LSTMs. By introducing a dummy value $y_0 = 0$, we do not even require to observe any points of the learning curve at test time.

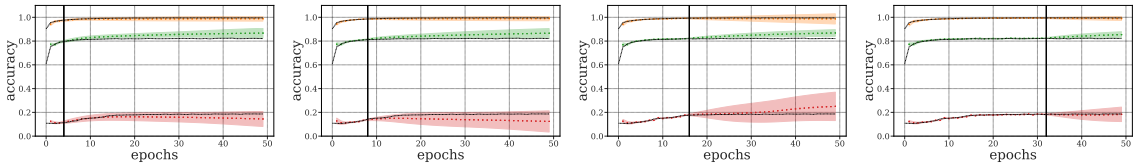
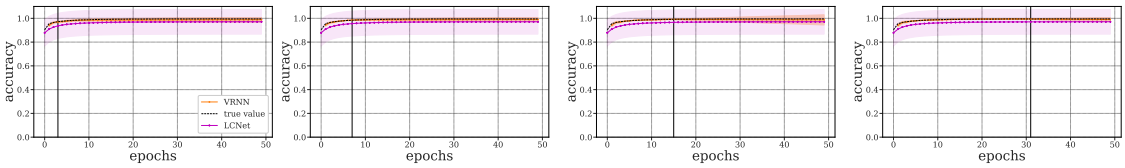


Figure 2: Qualitative assessment of the test roll-out performances of VRNN for different numbers of observed epochs (black vertical line) on the MNIST benchmark. Different colors stand for different configurations. The shaded area corresponds to 1σ.



(a) 4 observed epochs (b) 8 observed epochs (c) 16 observed epochs (d) 32 observed epochs

Figure 3: Qualitative comparison of the test predictions of VRNN and LCNet models for different numbers of observed epochs (the black vertical line) for one learning curve randomly sampled from the MNIST dataset. The shaded area corresponds to 1σ.

3. Experiments

In this section, we first empirically evaluate the performances of our probabilistic models, dubbed VRNN and RF, and compare them against other non roll-out state-of-the-art probabilistic regression models for learning curve prediction. Afterwards, we present some preliminary results that show the potential of our model to predict learning curves from unseen datasets, and that hint towards possible meta-learning and transfer-learning extensions of this work.

3.1 Learning Curve Prediction

To test the predictive strength of our learning curve model, we generated four different sets of learning curves of a feed forward neural network as training data. For each dataset, we sampled 5000 hyperparameter configurations randomly from the configuration space described in Table 1 in Section B in the Appendix and trained each configuration for 50 epochs with Adam (Kingma and Welling, 2014) on the datasets MNIST (LeCun et al., 2001), Adult (Kohavi, 1996), Higgs (Baldi et al., 2014) and Vehicle (Siebert, 1987) collected from OpenML (Vanschoren et al., 2014). The same learning curve datasets were also used in (Falkner et al., 2018). For our experiments we used 25% of each dataset as test data, and trained each method on the remaining part with full-length learning curves. Due to space constraints, we only show experiments on the MNIST benchmark (see the Appendix for the experiments with the other datasets).

As baselines, we consider LCNet (Klein et al., 2017b), a random forest baseline (RF-B) as described by Klein et al. (2017b) and the last seen value (LSV) heuristic, that, despite its simplicity, is successfully used in Hyperband (Li et al., 2017). Note that, LSV does not provide uncertainty estimates, therefore we used it only to compare against mean predictions. While we found the random forest based methods to be robust against their own

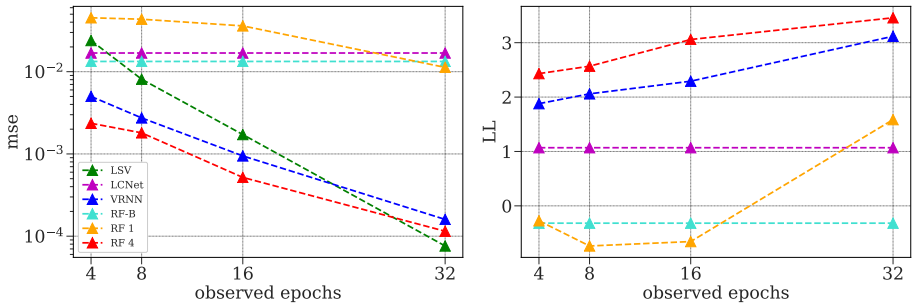


Figure 4: Assessment of the test predictive quality of the different models at target epoch 40 on the MNIST benchmark for different numbers of observed points from the learning curves. Note that RF-B and LCNet are not capable of adapting their predictions online without retraining, hence their constant error across epochs. Note that LSV does not provide a predictive variance.

hyperparameters, we used BOHB (Falkner et al., 2018) to optimize the hyperparameters of our model and LCNet on the MNIST dataset and then used the best found configuration for all experiments (see Section C in the Appendix for more details).

Even though LCNet and RF-B allow to predict for completely unobserved curves, only our models are able to correct their test predictions on the fly without the need for retraining after observing initial points from the true learning curve. This property, illustrated in Figures 2, 3 and 4, (see Section E in the Appendix for additional Figures), is fundamental for multi-fidelity hyperparameter optimization methods, such as Hyperband (Li et al., 2017) or BOHB (Falkner et al., 2018), where learning curves of different configurations are extended to different budgets. In particular, in Figures 3 and 4, the test performances of different methods for different numbers of observed points at test time are shown. All the models were trained on the MNIST benchmark for full-length learning curves. The roll-out methods take as input also the extra information provided at test time from the partially observed learning curves. Therefore their predictions do not remain unchanged, as those produced by LCNet and RF-B models, but improve with increasing number of observed epochs.

This flexibility comes together with a higher quality of predictions, as shown in Figures 3 and 5 (see also the Tables in Section D of the Appendix). In addition, another benefit compared to the LCNet model is that our model does not rely on prior user knowledge of the learning curves shape through the use of the parametric basis functions.

As shown in Figure 4, the performance of roll-out models based on random forests degrades significantly with the reduction of the input size (see also Figures 12–19, 21 and 22 in Appendix E). This is due to their intrinsic inability of modeling sequence-type data, such as learning curves. Therefore, this class of methods might require an input size whose cost could realistically be non-negligible for scenarios such as learning curves generated by the training of state-of-the-art deep neural networks.

3.2 Predictions for Unseen Datasets

We now conduct preliminary experiments to study the performances of our roll-out models on unseen datasets, by training them on the MNIST benchmark and using these trained models to extrapolate partial learning curves on other datasets without retraining. Even

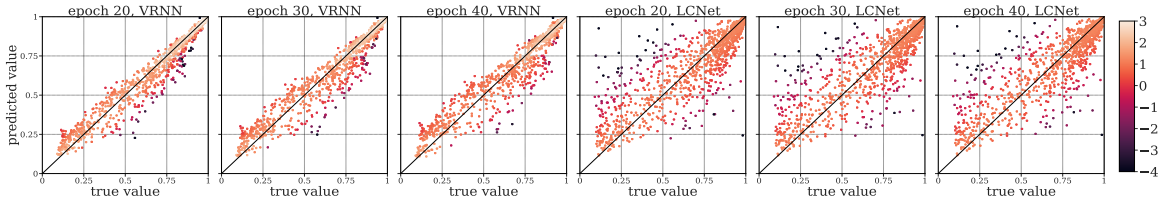


Figure 5: Qualitative assessment at different target epochs on the MNIST benchmark of the test roll-out performances of VRNN with 4 observed epochs and LCNet. Each plot shows on the horizontal axis the true values and on the vertical axis the predicted values. Each point is colored based on its log-likelihood value.

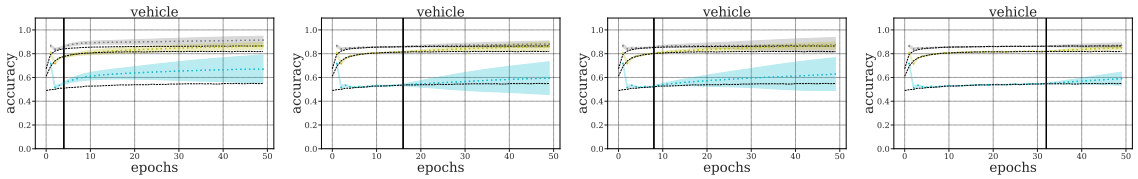


Figure 6: Qualitative assessment of VRNN predictions on the Vehicle benchmark when trained on MNIST for different numbers of observed epochs (black vertical line).

though the same configuration potentially leads to vastly different performances across different datasets (as also observed in our benchmarks), Figure 6 suggests that the VRNN model can adjust its predictions on the fly by starting its roll-outs with the initial learning curves observed on the new dataset. Based on these results (see also Figures 26–33 in Appendix E), we believe that our model is very promising for a variety of meta-learning and transfer-learning extensions.

4. Conclusion and Possible Use Cases of our Roll-Out Models

We proposed new roll-out models for the learning curve prediction task, based on random forests and variational recurrent neural networks. These models offer more flexibility and better performances than previous state-of-the-art learning curve prediction methods from the literature. In addition, they show to be capable of adapting their predictions to unseen datasets. We now list some of the possible future extensions of this work in AutoML tasks:

- Explicit dataset meta-features can be integrated and/or a latent task embedding learnt in order to enable direct learning across datasets.
- Our model could be used to warmstart bandit-based hyperparameter optimizers, such as Hyperband and BOHB, and replace the individual models learnt by BOHB for each fidelity.
- Our model could also be used to directly make decisions about which learning curves to extend, akin to Freeze-Thaw Bayesian optimization (Swersky et al., 2014).
- High-quality and flexible uncertainty estimates of time-series predictions are important in scenarios based on the exploitation-exploration paradigm. This makes our model attractive also for reinforcement-learning applications.

Due to this breadth of possible use cases, we expect our model to be useful in developing different types of new AutoML systems. To facilitate this, we make open-source code available for our model and experiments at `https://github.com/gmatilde/vdrnn`.

Acknowledgments

This work has partly been supported by Robert Bosch GmbH, the state of Baden-Württemberg through bwHPC and the German Research Foundation (DFG) through grant no INST 39/963-1 FUGG, and the European Research Council (ERC) under the European Unions Horizon 2020 research and innovation programme under grant no. 716721.

References

B. Baker, O. Gupta, R. Raskar, and N. Naik. Accelerating neural architecture search using performance prediction. *arXiv:1705.10823 [cs.LG]*, 2017.

R. Baldi, P. Sadowski, and D. Whiteson. Searching for exotic particles in high-energy physics with deep learning. *Nature communications*, 2014.

Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *Proceedings of the 34th International Conference on Machine Learning (ICML'09)*, 2019.

L. Breimann. Random forests. *Machine Learning*, 2001.

Y. Chen, A. Huang, Z. Wang, I. Antonoglou, J. Schrittwieser, D. Silver, and N. de Freitas. Bayesian optimization in AlphaGo. *arXiv:1812.06855 [cs.LG]*, 2018.

T. Domhan, J. T. Springenberg, and F. Hutter. Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI'15)*, 2015.

S. Falkner, A. Klein, and F. Hutter. BOHB: Robust and efficient hyperparameter optimization at scale. In *Proceedings of the 35th International Conference on Machine Learning (ICML 2018)*, 2018.

M. Feurer and F. Hutter. Hyperparameter optimization. In *Automatic Machine Learning: Methods, Systems, Challenges*. Springer, 2018.

Y. Gal and Z. Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of the 33rd International Conference on Machine Learning (ICML'16)*, 2016a.

Y. Gal and Z. Ghahramani. A theoretically grounded application of dropout in recurrent neural networks. In *Proceedings of the 28th International Conference on Advances in Neural Information Processing Systems (NIPS'16)*, 2016b.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 1997.

F. Hutter, L. Xu, H. Hoos, and K. Leyton-Brown. Algorithm runtime prediction: Methods and evaluation. *Artificial Intelligence*, 2014.

F. Hutter, L. Kotthoff, and J. Vanschoren, editors. *Automatic Machine Learning: Methods, Systems, Challenges*. Springer, 2018.

K. Kandasamy, G. Dasarathy, J. Schneider, and B. Póczos. Multi-fidelity bayesian optimisation with continuous approximations. In *Proceedings of the 34th International Conference on Machine Learning (ICML'17)*, 2017.

D. Kingma and M. Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations (ICLR'14)*, 2014.

A. Klein, S. Falkner, S. Bartels, P. Hennig, and F. Hutter. Fast Bayesian hyperparameter optimization on large datasets. In *Electronic Journal of Statistics*, 2017a.

A. Klein, S. Falkner, J. T. Springenberg, and F. Hutter. Learning curve prediction with Bayesian neural networks. In *International Conference on Learning Representations (ICLR'17)*, 2017b.

R. Kohavi. Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD'96)*, 1996.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Intelligent Signal Processing*. IEEE Press, 2001.

L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar. Hyperband: Bandit-based configuration evaluation for hyperparameter optimization. In *International Conference on Learning Representations (ICLR'17)*, 2017.

G. Melis, C. Dyer, and P. Blunsom. On the state of the art of evaluation in neural language models. In *International Conference on Learning Representations (ICLR'18)*, 2018.

S. S. Rangapuram, M. W. Seeger, J. Gasthaus, L. Stella, Y. Wang, and T. Januschowski. Deep state space models for time series forecasting. In *Proceedings of the 31th International Conference on Advances in Neural Information Processing Systems (NIPS'18)*. 2018.

J. P. Siebert. *Vehicle Recognition Using Rule Based Methods*. Turing Institute, 1987.

K. Swersky, J. Snoek, and R. Adams. Freeze-thaw Bayesian optimization. *arXiv:1406.3896 [stat.ML]*, 2014.

J. Vanschoren, J. van Rijn, B. Bischl, and L. Torgo. OpenML: Networked science in machine learning. *SIGKDD Explorations*, 2014.

Appendix A. Details about the models

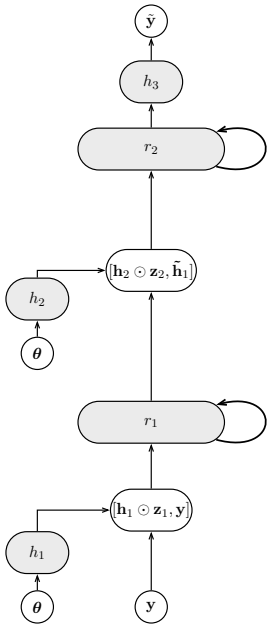


Figure 7: Folded schematic of the VRNN model for learning curve prediction. h_1, h_2 and h_3 are feedforward neural networks, r_1 and r_2 are LSTM blocks and $\tilde{\mathbf{h}}_1$ is the output of r_1. Given the learning curve $\tilde{\mathbf{y}}^{(\mathbf{i})} = (y_0, \ldots, y_T)$, in the graph $\tilde{\mathbf{y}} = (\tilde{y}_1, \ldots, \tilde{y}_T)$ is the vector of the predicted values, $\mathbf{y} = (y_0, y_1 \ldots, y_{T-1})$ is the input at training time and $\mathbf{y} = (y_0, \ldots \tilde{y}_{M-1}, \tilde{y}_M, \ldots, \tilde{y}_{T-1})$ is the input at evaluation time and where M is the number of observed points, $\mathbf{z_1}$ and $\mathbf{z_2}$ are dropout masks sampled from a Bernoulli distribution and are kept fixed across time steps, and $\boldsymbol{\theta}$ is the configuration vector, which is fed into a feedforward neural network and then used to initialize the hidden states of the LSTM blocks. The bold arrows indicate the recurrence.

Appendix B. Datasets

Table 1 reports the hyperparameters of the benchmarks from which the learning curves datasets described in Section 3.1 are generated.

Hyperparameter Name	Range	Log-Scale
initial learning rate	$[10^{-6}, 10^{-2}]$	✓
batch size	[16, 256]	✓
average units per layer	$[2^4, 2^8]$	✓
final learning rate fraction	$[10^{-4}, 10^{0}]$	✓
shape parameter 1	$[0, 1]$	✓
dropout 0	$[0.0, 0.5]$	–
dropout 1	$[0.0, 0.5]$	–
number of layers	$[1, 5]$	–

Table 1

Appendix C. Hyperparameter optimization

In order to select the architecture for our model and the hyperparameters that control the training procedure (see Table 3 for a list of the hyperparameters), we used BOHB (Falkner et al., 2018) as hyperparameter optimizer on the MNIST learning curves benchmark and the set-up described in Table 2. The configuration returned as incumbent was then used in all the VRNN experiments and showed good performances across all the considered datasets. The selected configuration is reported in Table 4.

In order to optimize the LCNet's hyperparameters, we also run BOHB with the same set-up (see Table 2) on MNIST learning curves benchmark. Since numerical instability problems were occurring during the training procedure when the incumbent configuration returned by BOHB was applied, for the experiments with this model we then used the default configuration, which appeared to be more robust (Table 5 reports a list of the hyperparameters of LCNet together with their default values).

As optimizers we used SGD with momentum and adaptive SGHMC for the VRNN and LCNet experiments, respectively. Regarding the experiments with the VRNN model, in order to speed up the training procedure, we also used a curriculum learning based technique (Bengio et al., 2019) and linearly increased the length of the input sequence during training, starting from a selected number of initial observed epochs (this hyperparameter, dubbed "initial observed epochs", was also optimized with BOHB and the selected value is reported in Table 4).

Hyperparameter Name	Value
η	2
number of iterations	1000
min time budget (min)	2
max time budget (min)	10

Table 2: Set-up of BOHB optimizer used to optimize VRNN and LCNet's hyperparameters.

Hyperparameter Name	Range	Log-Scale	Type
initial learning rate	$[10^{-5}, 10^{-1}]$	✓	FLOAT
momentum	$[0, 0.99]$	–	FLOAT
final learning fraction	$[10^{-4}, 10^{0}]$	✓	FLOAT
batch size	[4, 128]	✓	INTEGER
initial observed epochs	$[5, 50]$	✓	INTEGER
number of stacked LSTMs	$[1, 2]$	–	INTEGER
number of layers for final MLP	$[1, 2]$	–	INTEGER
number of layers for config. MLP	$[1, 2]$	–	INTEGER
number of units for LSTM	$[2^2, 2^7]$	✓	INTEGER
number of units for final MLP	$[2^2, 2^7]$	✓	INTEGER
number of units for config. MLP	$[2^2, 2^7]$	✓	INTEGER
learning rate scheduler	[cos, exp, const]	–	CATEGORICAL

Table 3: Hyperparameter configuration space of the VRNN model described in Section 2.2.

Hyperparameter Name	Selected Value
initial learning rate	0.027
final learning fraction	0.0008
batch size	22
initial observed epochs	5
number of stacked LSTMs	2
number of layers for final MLP	1
number of layers for config. MLP	1
number of units for LSTM	6
number of units for final MLP	103
number of units for config. MLP	115
learning rate scheduler	cos

Table 4: Incumbent hyperparameter configuration selected by BOHB optimizer.

Hyperparameter Name	Range	Log-Scale	Type	Default
learning rate	$[10^{-5}, 10^{-1}]$	✓	FLOAT	0.001
momentum	$[0, 0.99]$	–	FLOAT	0.05
batch size	$[4, 128]$	✓	INTEGER	40

Table 5: Hyperparameter configuration space of LCNet.

Appendix D. Tables with Mean Squared Error and Median Log-Likelihood

Tables 6–9 report for each method the achieved mean squared error and median log-likelihood averaged over all the epochs on the four considered datasets for different numbers of observed epochs at test time, respectively. The same metrics but across different epochs are also plotted in Figures 24 and 25. We observe that our VRNN model and RF 4 consistently yield to better predictions. VRNN* and RF* 4 are used in the Tables to denote respectively VRNN and RF 4 models trained on the MNIST benchmark and then used to evaluate on the other unseen datasets.

epochs 4	mnist		higgs		adult		vehicle	
Methods	**mse**	**ll**	**mse**	**ll**	**mse**	**ll**	**mse**	**ll**
VRNN	3e−3 ± 0.01	2.2 ± 1.1	6e−4 ± 0.0	2.75 ± 1.8	1e−3 ± 0.01	2.57 ± 0.9	1e−3 ± 0.0	2.9 ± 1.3
RF 1	0.03 ± 0.07	0.89 ± 927	5e−4 ± 0.0	2.9 ± 5.5	1e−3 ± 0.01	3.7 ± 8.6	2e−3 ± 0.01	3.4 ± 16
RF 4	0.02 ± 0.01	2.7 ± 4.5	2e−4 ± 0.0	3.2 ± 0.9	3e−4 ± 0.00	3.8 ± 1.12	4e−4 ± 0.0	3.9 ± 1.12
LSV	0.02 ± 0.03	−−	1e−3 ± 0.0	−−	3e−3 ± 0.01	−−	4e−3 ± 0.01	−−
RF-B	0.01 ± 0.05	0.01 ± 1.0	5e−4 ± 0.0	1.6 ± 0.4	2e−3 ± 0.01	1.6 ± 1.1	1e−3 ± 0.0	1.3 ± 0.9
LCNet	0.02 ± 0.04	1.03 ± 1.0	3e−3 ± 0.0	1.21 ± 0.16	0.01 ± 0.02	1.12 ± 0.74	5e−3 ± 0.01	1.41 ± 0.6
VRNN*	3e−3 ± 0.01	2.2 ± 1.1	0.01 ± 0.01	−2.7 ± 3.4	3e−3 ± 0.0	0.86 ± 1.94	3e−3 ± 0.0	1.16 ± 2.3
RF* 4	0.02 ± 0.01	2.7 ± 4.5	1e−3 ± 0.0	2.16 ± 1.37	1e−3 ± 0.0	2.15 ± 1.3	1e−3 ± 0.0	2.37 ± 1.15

Table 6: Average total mean squared error and median log-likelihood achieved by the different models for 4 observed epochs at evaluation time.

epochs 8	mnist		higgs		adult		vehicle	
Methods	**mse**	**ll**	**mse**	**ll**	**mse**	**ll**	**mse**	**ll**
VRNN	2e−3±0.01	2.53 ± 2.1	2e−4 ± 0.0	3 ± 1.5	5e−4 ± 0.0	2.9 ± 0.8	5e−4 ± 0.0	3.25 ± 1.1
RF 1	0.02 ± 0.06	0.68 ± 243	4e−4 ± 0.0	3.2 ± 5.5	1e−3±0.01	3.9 ± 8.6	1e−3±0.01	3.7 ± 13.5
RF 4	1e−3±0.05	2.88 ± 3.9	2e−4 ± 0.0	3.32 ± 0.73	2e−4 ± 0.0	3.94 ± 1.1	3e−4±0.10	3.96 ± 1.1
LSV	5e−3 ± 0.1	−−	5e−4 ± 0.0	−−	8e−4±0.01	−−	1e−3 ± 0.0	−−
RF-B	0.01 ± 0.05	0.01 ± 1.0	5e−4 ± 0.0	1.6 ± 0.4	2e−3±0.01	1.6 ± 1.1	1e−3 ± 0.0	1.3 ± 0.9
LCNet	0.02 ± 0.04	1.03 ± 1.0	3e−3 ± 0.0	1.21 ± 0.16	0.01 ± 0.02	1.12 ± 0.74	5e−3±0.01	1.41 ± 0.6
VRNN*	2e−3±0.01	2.53 ± 2.1	3e−3 ± 0.0	−0.78±2.4	1e−3 ± 0.0	1.75 ± 1.34	1e−3 ± 0.0	2.3 ± 2.1
RF* 4	1e−3±0.05	2.88 ± 3.9	1e−3 ± 0.0	2.5 ± 0.6	7e−3 ± 0.0	2.52 ± 1.0	1e−3 ± 0.0	2.65 ± 1.26

Table 7: Average total mean squared error and median log-likelihood achieved by the different models for 8 observed epochs at evaluation time.

epochs 16	**mnist**		**higgs**		**adult**		**vehicle**	
Methods	**mse**	**ll**	**mse**	**ll**	**mse**	**ll**	**mse**	**ll**
VRNN	5e−4 ± 0.0	3 ± 1.1	7e−5 ± 0.0	3.3 ± 0.9	2e−4 ± 0.0	3.38 ± 0.7	2e−4 ± 0.0	3.7 ± 1.3
RF 1	0.02 ± 0.04	1.13 ± 89.6	3e−4 ± 0.0	3.48 ± 2.9	1e−3 ± 0.0	4.1 ± 5.6	1e−3 ± 0.0	3.99 ± 7.8
RF 4	2e−4 ± 0.0	3.64 ± 1.2	6e−5 ± 0.0	3.81 ± 0.7	9e−5 ± 0.0	4.4 ± 0.8	2e−4 ± 0.0	4.47 ± 1.0
LSV	8e−4 ± 0.0	−−	1e−4 ± 0.0	−−	1e−4 ± 0.0	−−	3e−4 ± 0.0	−−
RF-B	0.01 ± 0.05	0.01 ± 1.0	5e−4 ± 0.0	1.6 ± 0.4	2e−3±0.01	1.6 ± 1.1	1e−3 ± 0.0	1.3 ± 0.9
LCNet	0.02 ± 0.04	1.03 ± 1.0	3e−3 ± 0.0	1.21 ± 0.16	0.01 ± 0.02	1.12 ± 0.74	5e−3±0.01	1.41 ± 0.6
VRNN*	5e−4 ± 0.0	3 ± 1.1	2e−3 ± 0.0	1e−3±2.35	7e−4 ± 0.0	2.35 ± 1.41	1e−3 ± 0.0	2.89 ± 2.1
RF* 4	2e−4 ± 0.0	3.64 ± 1.2	1e−3 ± 0.0	3.1 ± 0.6	3e−4 ± 0.0	3.2 ± 0.5	3e−4 ± 0.0	3.4 ± 1.2

Table 8: Average total mean squared error and median log-likelihood achieved by the different models for 16 observed epochs at evaluation time.

epochs 32	**mnist**		**higgs**		**adult**		**vehicle**	
Methods	**mse**	**ll**	**mse**	**ll**	**mse**	**ll**	**mse**	**ll**
VRNN	1e−4 ± 0.0	3.7 ± 1.2	2e−5 ± 0.0	3.81 ± 1.5	6e−5 ± 0.0	4.04 ± 0.8	4e−5 ± 0.0	4.49 ± 1.6
RF 1	5e−3±0.01	2.62 ± 3.7	1e−4 ± 0.0	3.97 ± 0.9	3e−4 ± 0.0	4.43 ± 1.8	4e−4 ± 0.0	4.64 ± 2.5
RF 4	4e−5 ± 0.0	4.31 ± 0.8	2e−5 ± 0.0	4.31 ± 0.3	3e−5 ± 0.0	4.8 ± 0.6	4e−5 ± 0.0	5.0 ± 1.3
LSV	3e−5 ± 0.0	−−	7e−6 ± 0.0	−−	8e−6 ± 0.0	−−	9e−6 ± 0.0	−−
RF-B	0.01 ± 0.05	0.01 ± 1.0	5e−4 ± 0.0	1.6 ± 0.4	2e−3±0.01	1.6 ± 1.1	1e−3 ± 0.0	1.3 ± 0.9
LCNet	0.02 ± 0.04	1.03 ± 1.0	3e−3 ± 0.0	1.21 ± 0.16	0.01 ± 0.02	1.12 ± 0.74	5e−3±0.01	1.41 ± 0.6
VRNN*	1e−4 ± 0.0	3.7 ± 1.2	1e−3 ± 0.0	0.54 ± 2.4	4e−4 ± 0.0	3.123±1.63	5e−4 ± 0.0	3.75 ± 2.9
RF* 4	4e−5 ± 0.0	4.31 ± 0.8	1e−4 ± 0.0	3.8 ± 0.3	7e−5 ± 0.0	4.03 ± 0.36	6e−5 ± 0.0	4.25 ± 0.6

Table 9: Average total mean squared error and median log-likelihood achieved by the different models for 32 observed epochs at evaluation time.

Appendix E. Additional Plots

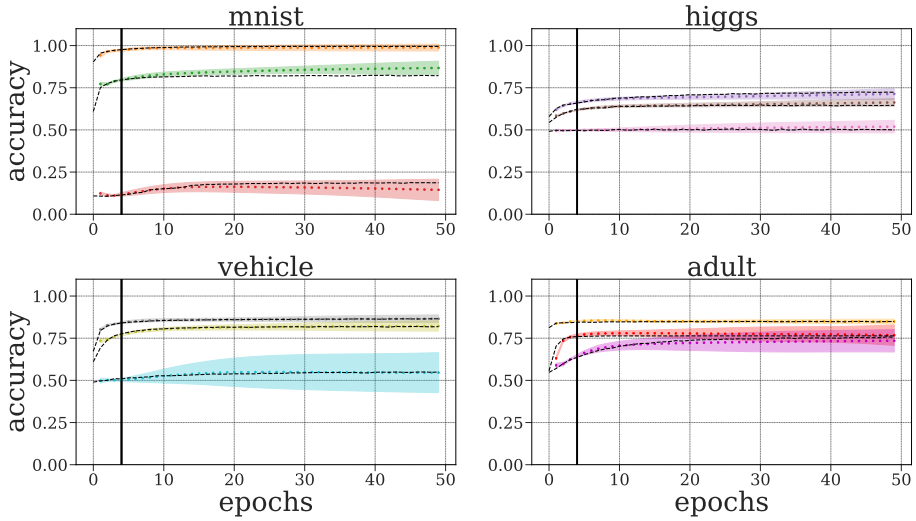


Figure 8: Qualitative assessment of the test roll-out performances of VRNN for 4 observed epochs (the black vertical line). Different colors of learning curves stand for different configurations, while the black dashed lines represent the true learning curves.

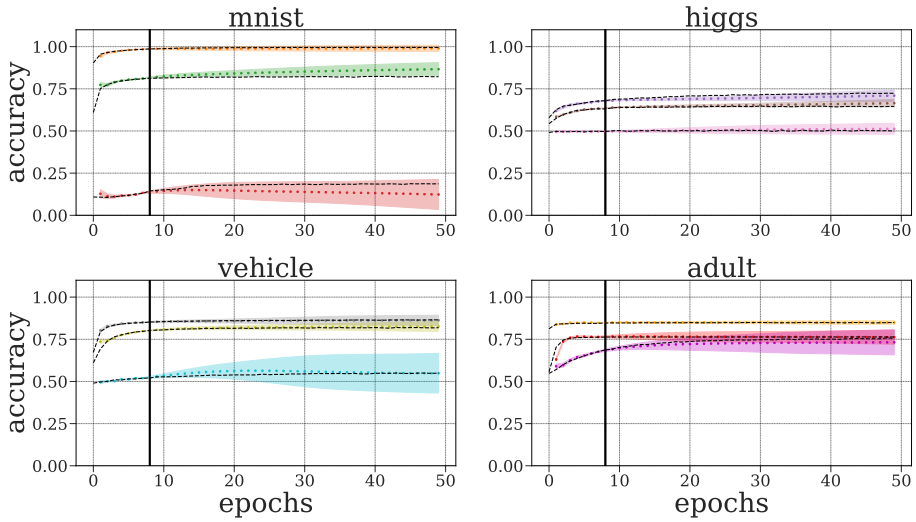


Figure 9: Qualitative assessment of the test roll-out performances of VRNN for 8 observed epochs (the black vertical line). Different colors of learning curves stand for different configurations, while the black dashed lines represent the true learning curves.

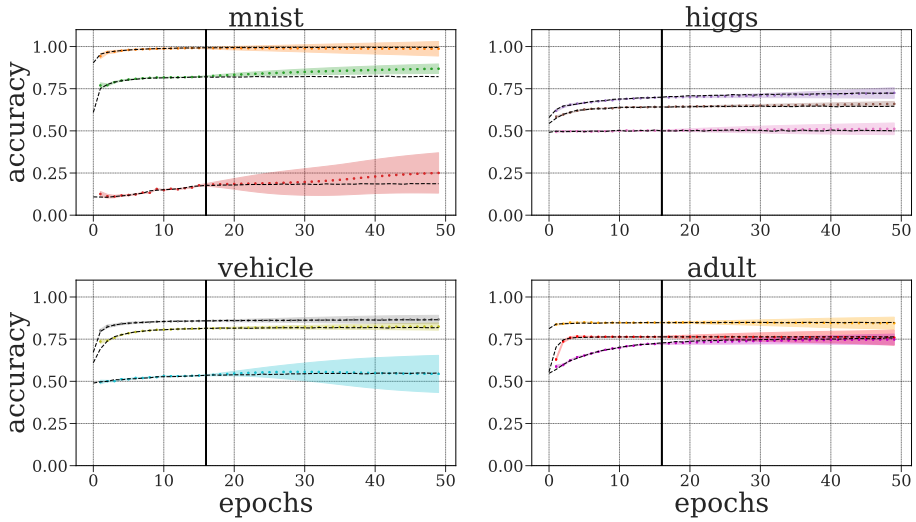


Figure 10: Qualitative assessment of the test roll-out performances of VRNN for 16 observed epochs (the black vertical line). Different colors of learning curves stand for different configurations, while the black dashed lines represent the true learning curves.

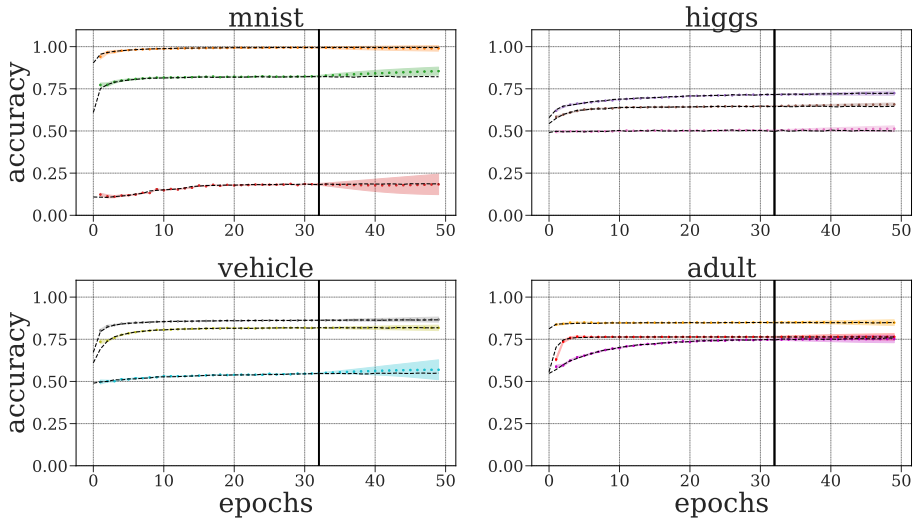


Figure 11: Qualitative assessment of the test roll-out performances of VRNN for 32 observed epochs (the black vertical line). Different colors of learning curves stand for different configurations, while the black dashed lines represent the true learning curves.

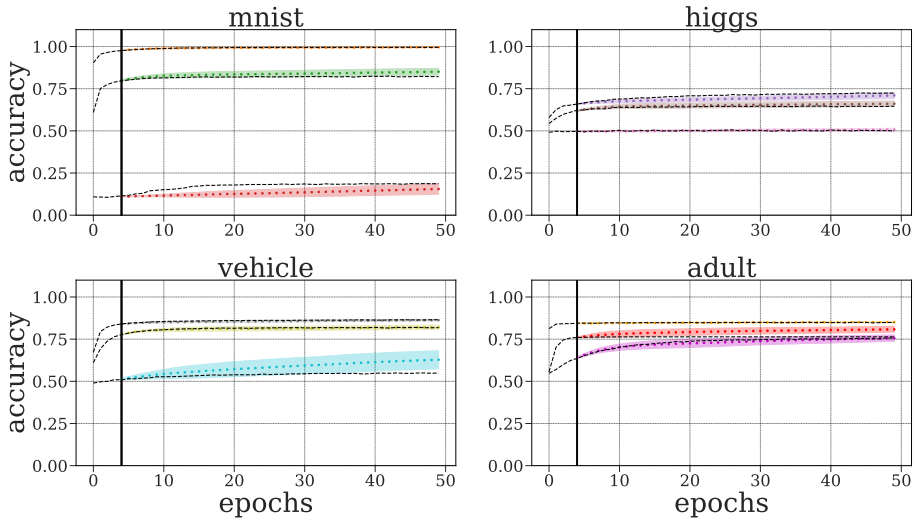


Figure 12: Qualitative assessment of the test roll-out performances of RF 4 for 4 observed epochs (the black vertical line). Different colors of the learning curves stand for different configurations, while the black dashed lines represent the true learning curves.

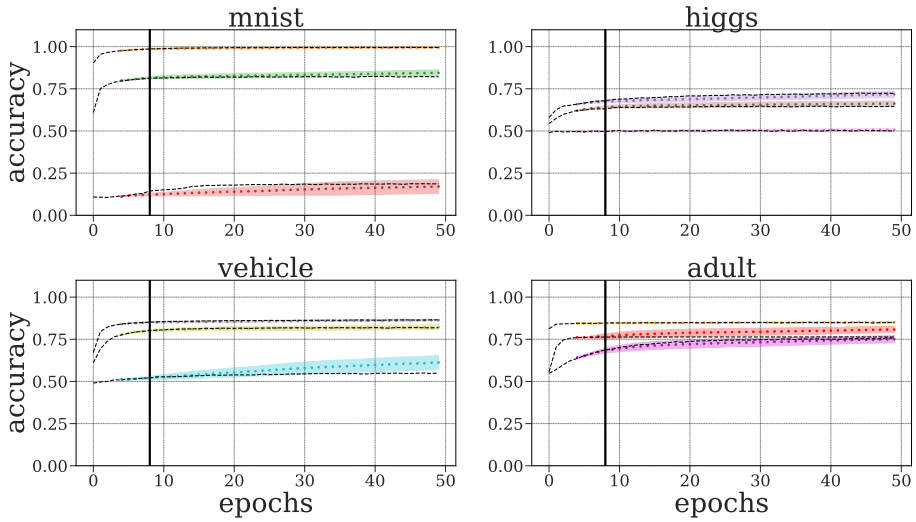


Figure 13: Qualitative assessment of the test roll-out performances of RF 4 for 8 observed epochs (the black vertical line). Different colors of the learning curves stand for different configurations, while the black dashed lines represent the true learning curves.

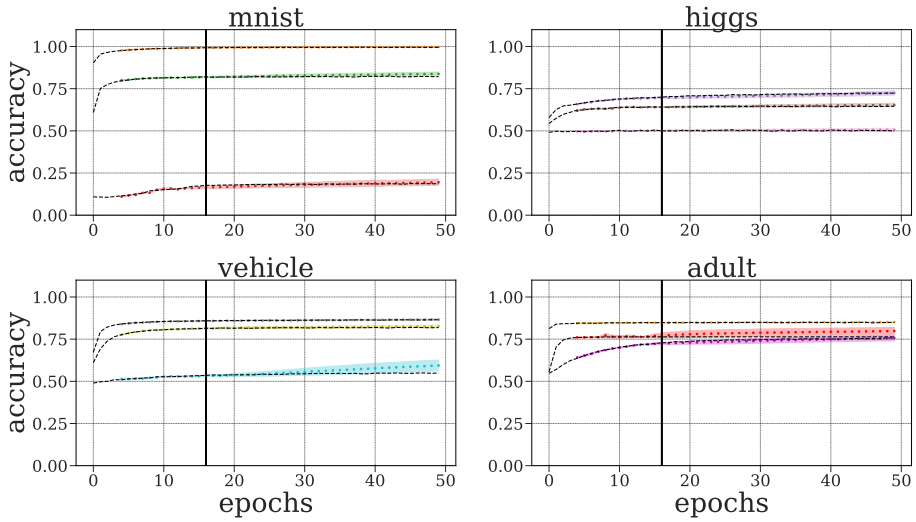


Figure 14: Qualitative assessment of the test roll-out performances of RF 4 for 16 observed epochs (the black vertical line). Different colors of the learning curves stand for different configurations, while the black dashed lines represent the true learning curves.

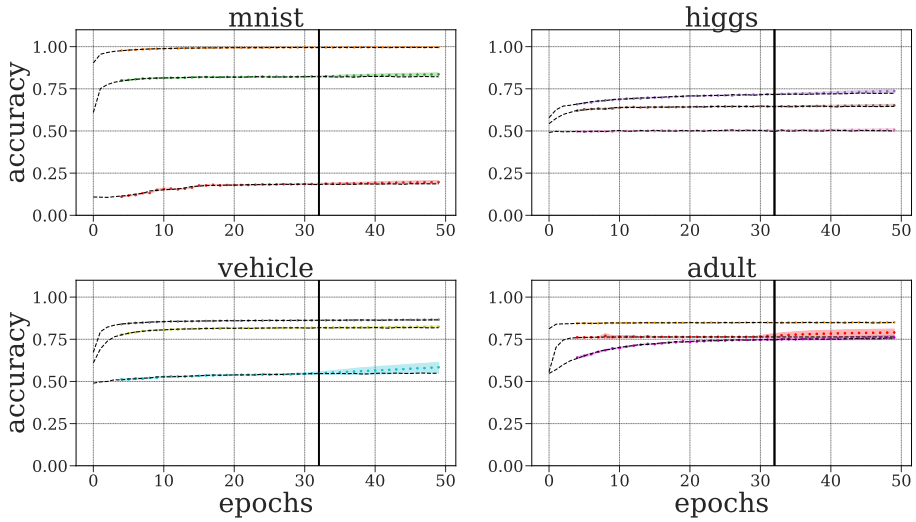


Figure 15: Qualitative assessment of the test roll-out performances of RF 4 for 32 observed epochs (the black vertical line). Different colors of the learning curves stand for different configurations, while the black dashed lines represent the true learning curves.

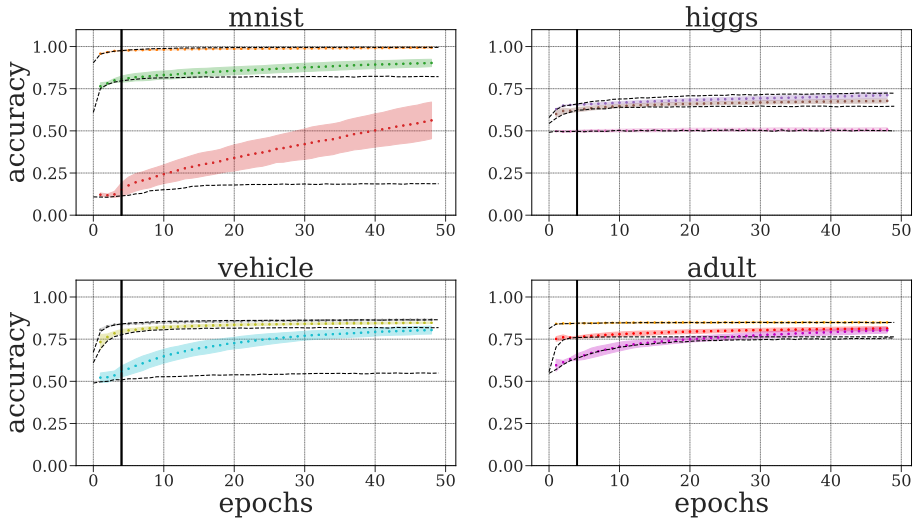


Figure 16: Qualitative assessment of the test roll-out performances of RF 1 for 4 observed epochs (the black vertical line). Different colors of the learning curves stand for different configurations, while the black dashed lines represent the true learning curves.

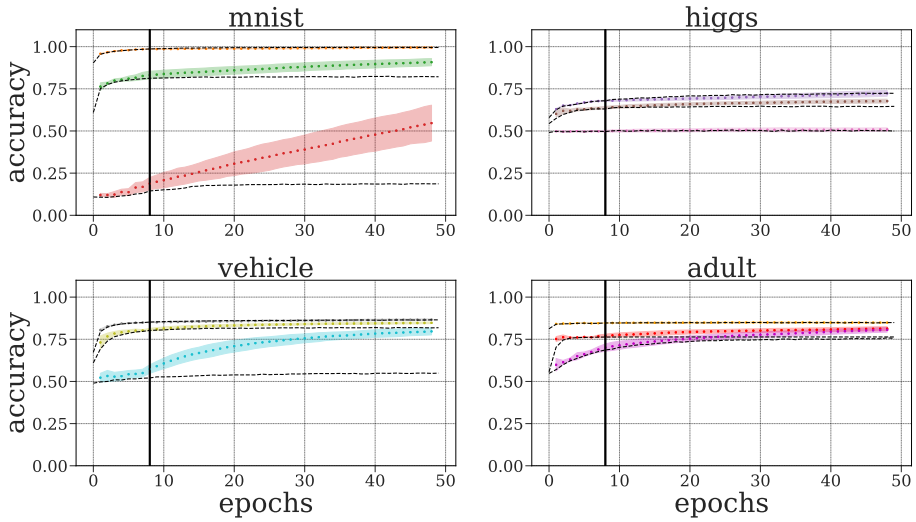


Figure 17: Qualitative assessment of the test roll-out performances of RF 1 for 8 observed epochs (the black vertical line). Different colors of the learning curves stand for different configurations, while the black dashed lines represent the true learning curves.

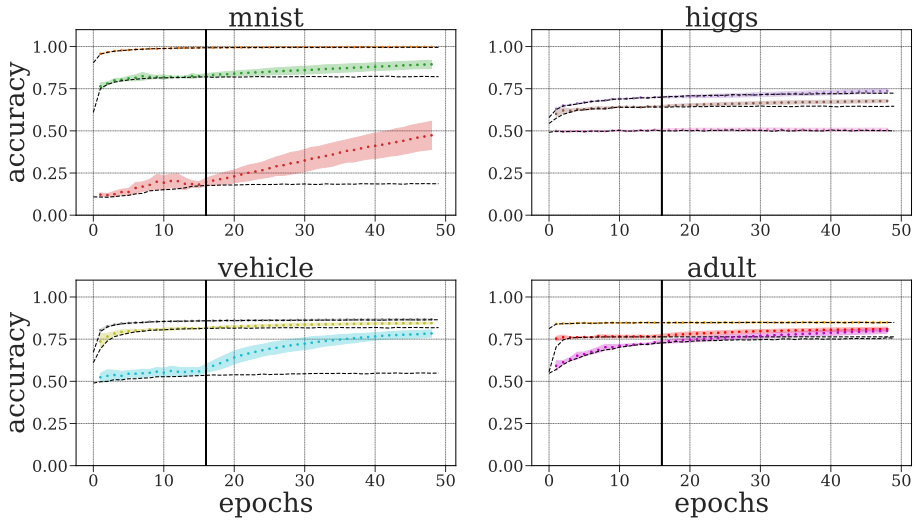


Figure 18: Qualitative assessment of the test roll-out performances of RF 1 for 16 observed epochs (the black vertical line). Different colors of the learning curves stand for different configurations, while the black dashed lines represent the true learning curves.

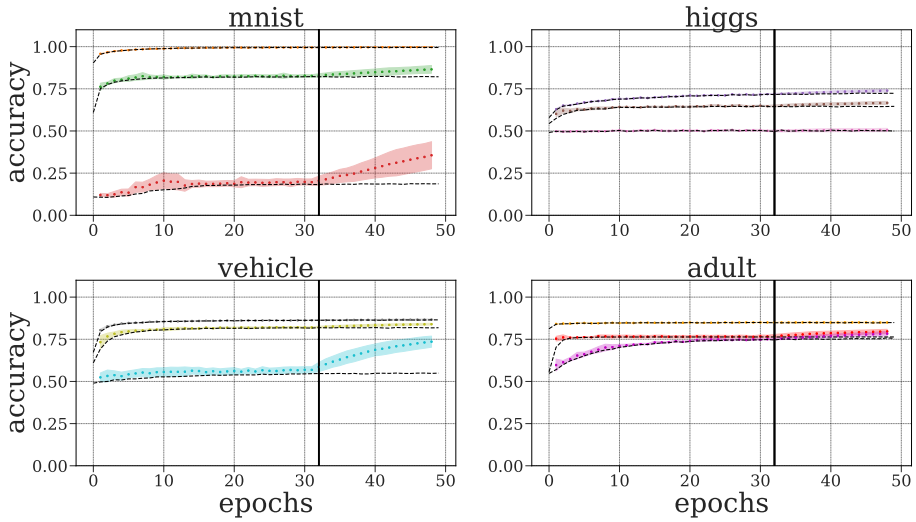


Figure 19: Qualitative assessment of the test roll-out performances of RF 1 for 32 epochs (the black vertical line). Different colors of the learning curves stand for different configurations, while the black dashed lines represent the true learning curves.

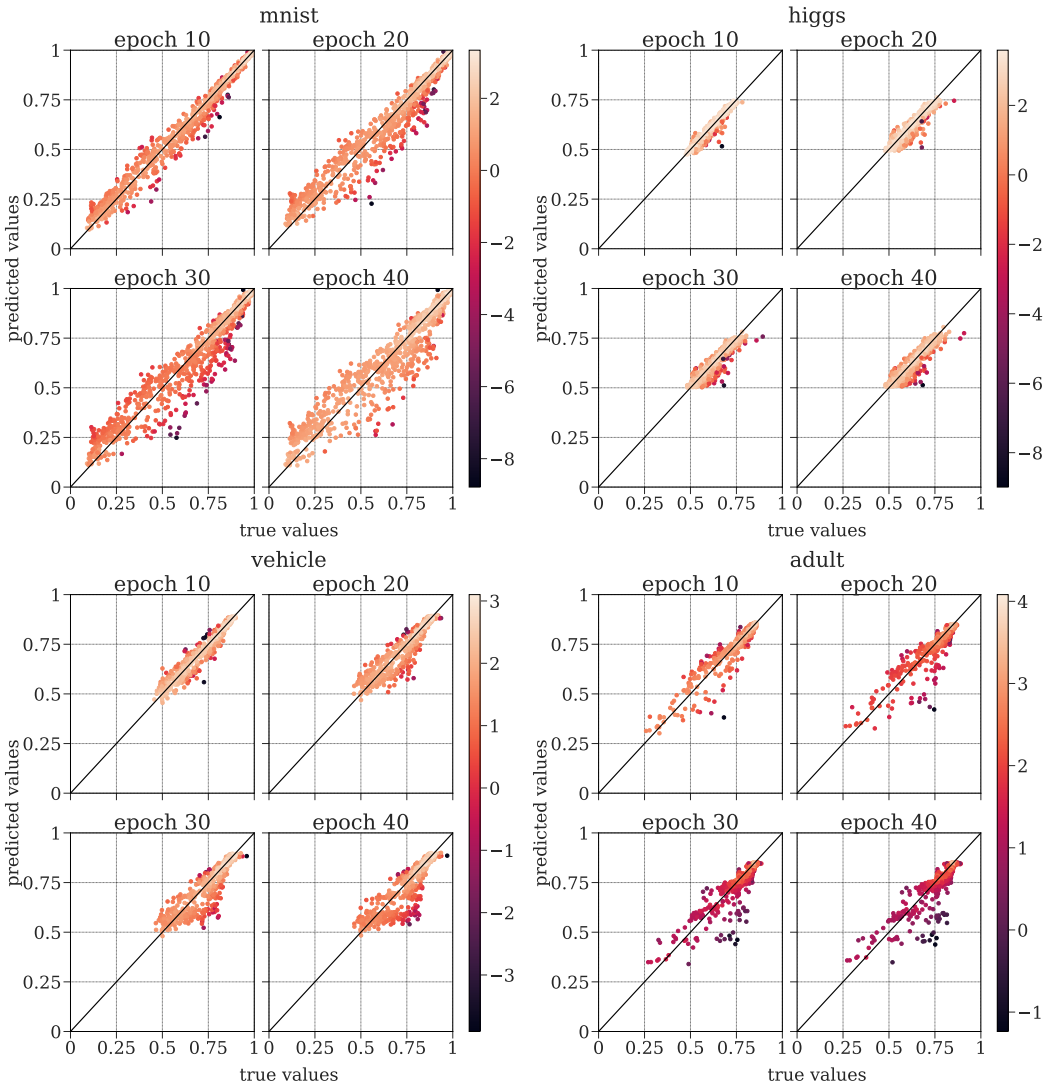


Figure 20: Qualitative assessment at different target epochs of the test roll-out performances of VRNN with 4 observed epochs on the four different datasets. Each plot shows on the horizontal axis the true values and on the vertical axis the predicted values. Each point is colored based on its log-likelihood value.

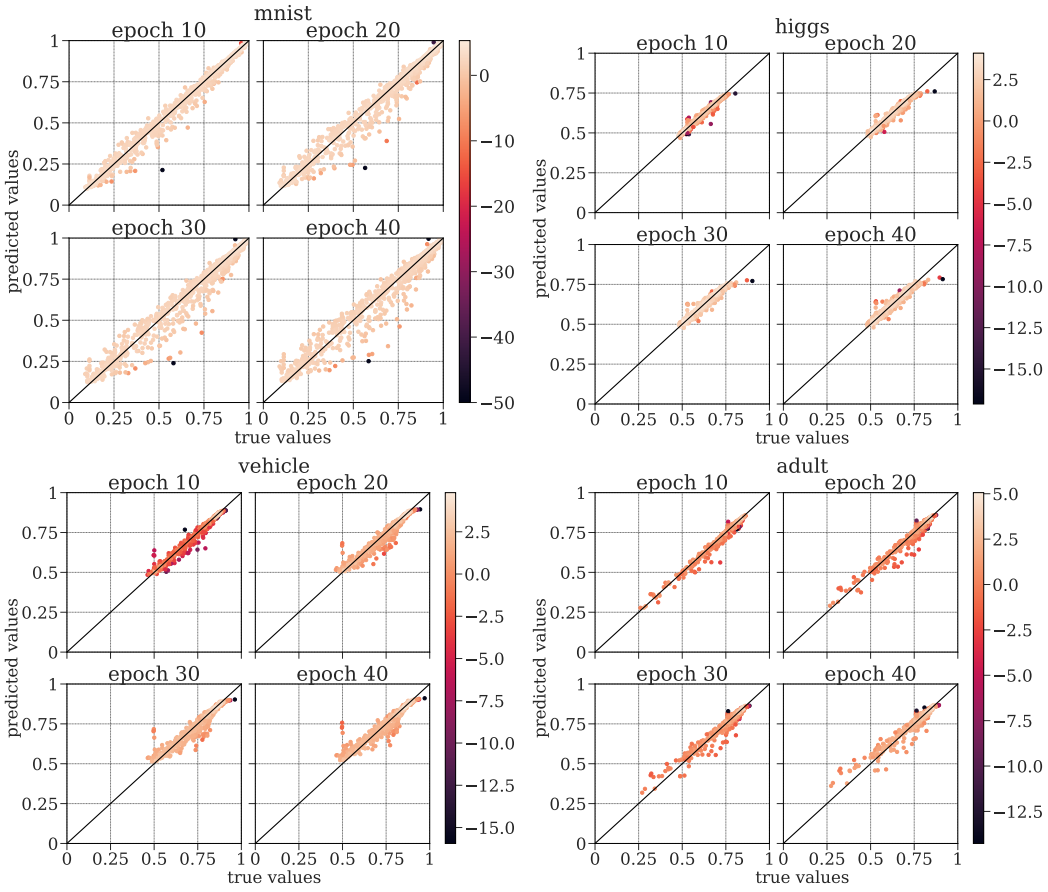


Figure 21: Qualitative assessment at different target epochs of the test roll-out performances of RF 4 with 4 observed epochs on the four different datasets. Each plot shows on the horizontal axis the true values and on the vertical axis the predicted values. Each point is colored based on its log-likelihood value.

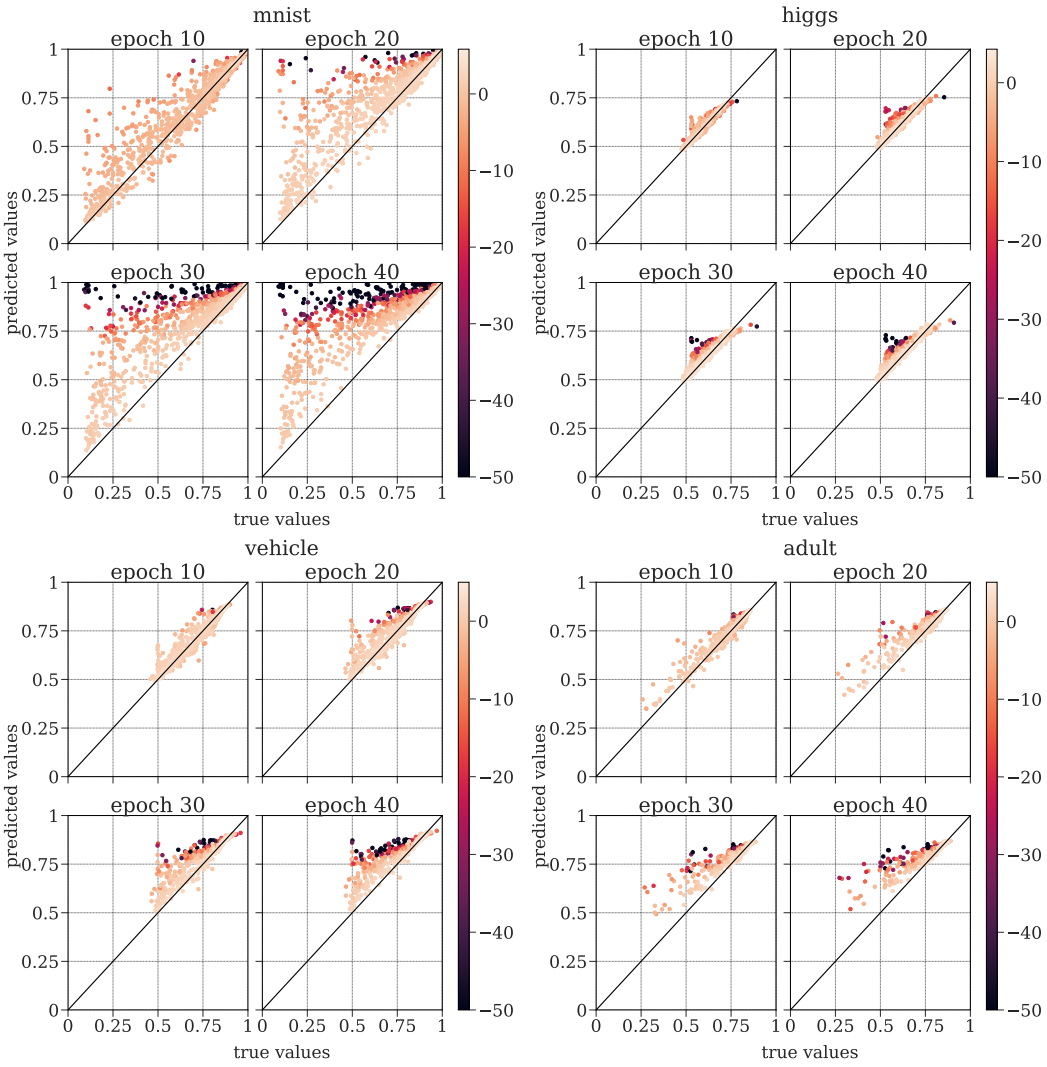
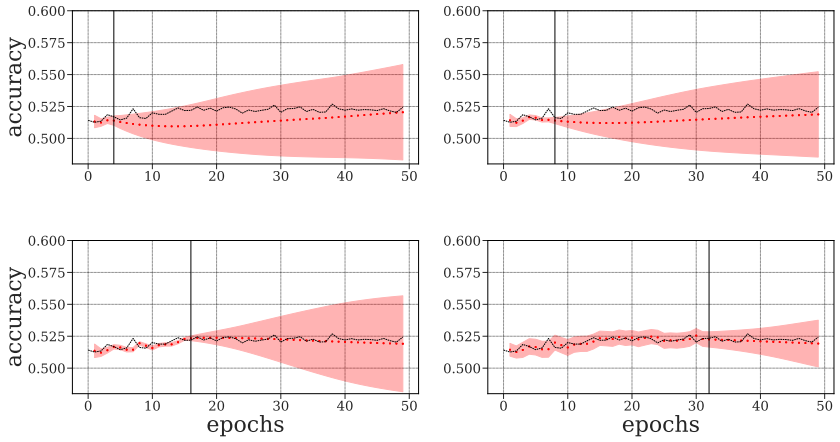
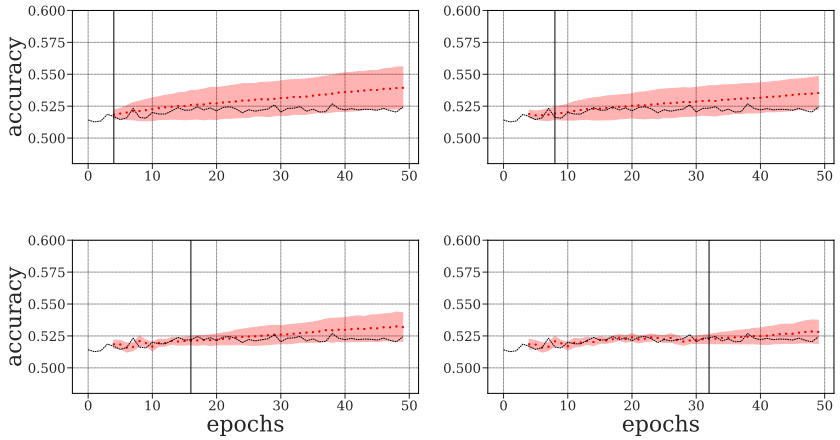


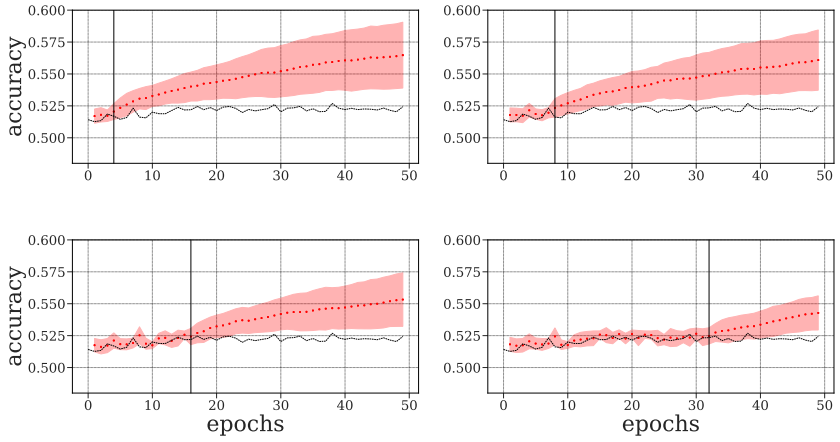
Figure 22: Qualitative assessment at different target epochs of the test roll-out performances of RF 1 with 4 observed epochs on the four different datasets. Each plot shows on the horizontal axis the true values and on the vertical axis the predicted values. Each point is colored based on its log-likelihood value.



(a) VRNN.



(b) RF 4.



(c) RF 1.

Figure 23: Predictions of roll-out models for the case of a very bumpy learning curve from the Higgs benchmark.

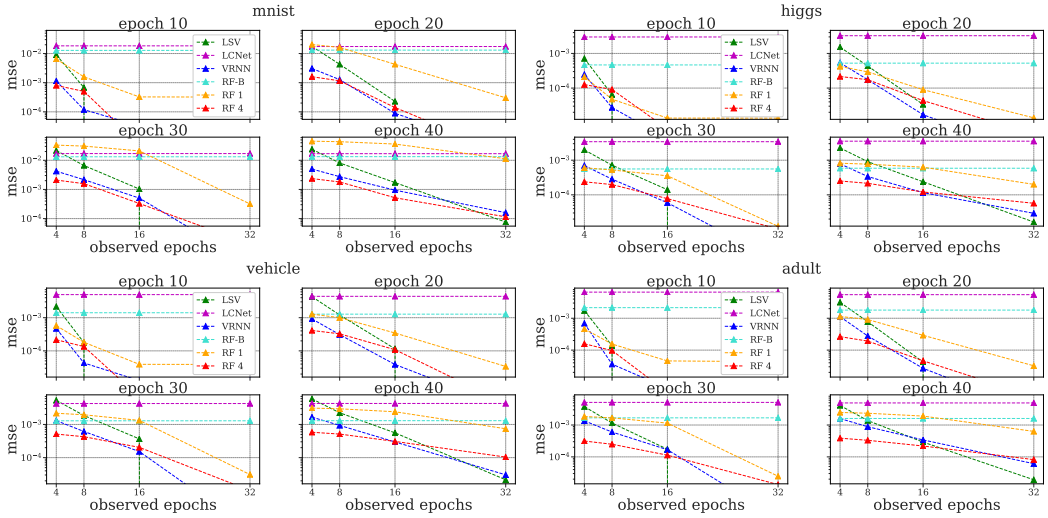


Figure 24: The panels show how the mean squared error at different target epochs (y-axis) varies with the number of observed points from the true learning curve at evaluation time (x-axis) for different methods on the four considered benchmarks.

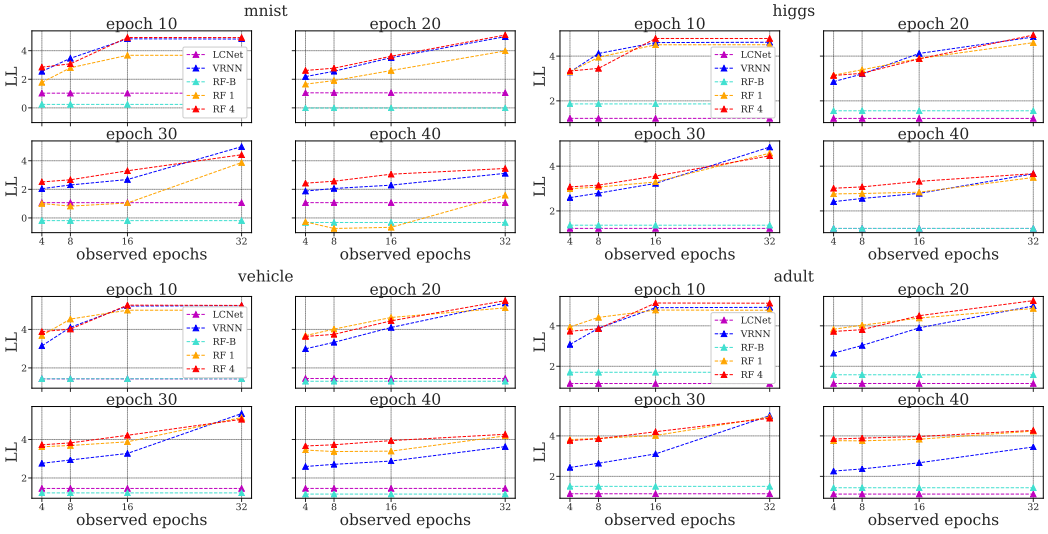


Figure 25: The panels show how the median log-likelihood at different target epochs (y-axis) varies with the number of observed points from the true learning curve at evaluation time (x-axis) for different methods on the four considered benchmarks.

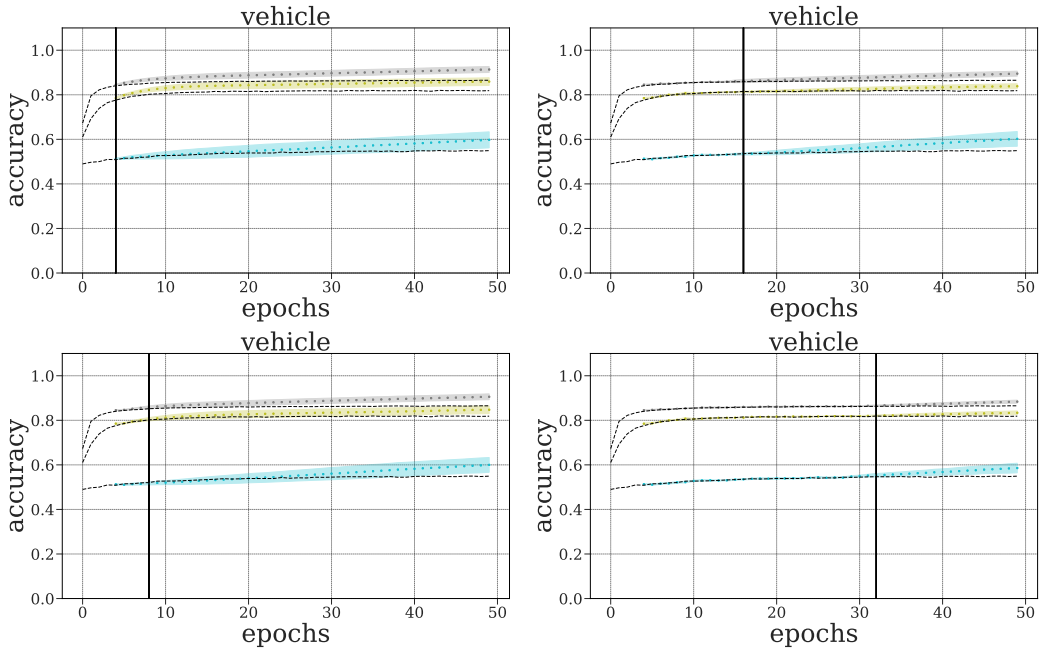


Figure 26: Qualitative assessment of RF 4 predictions on the Vehicle benchmark when trained on MNIST for different numbers of observed epochs at test time (the black vertical line).

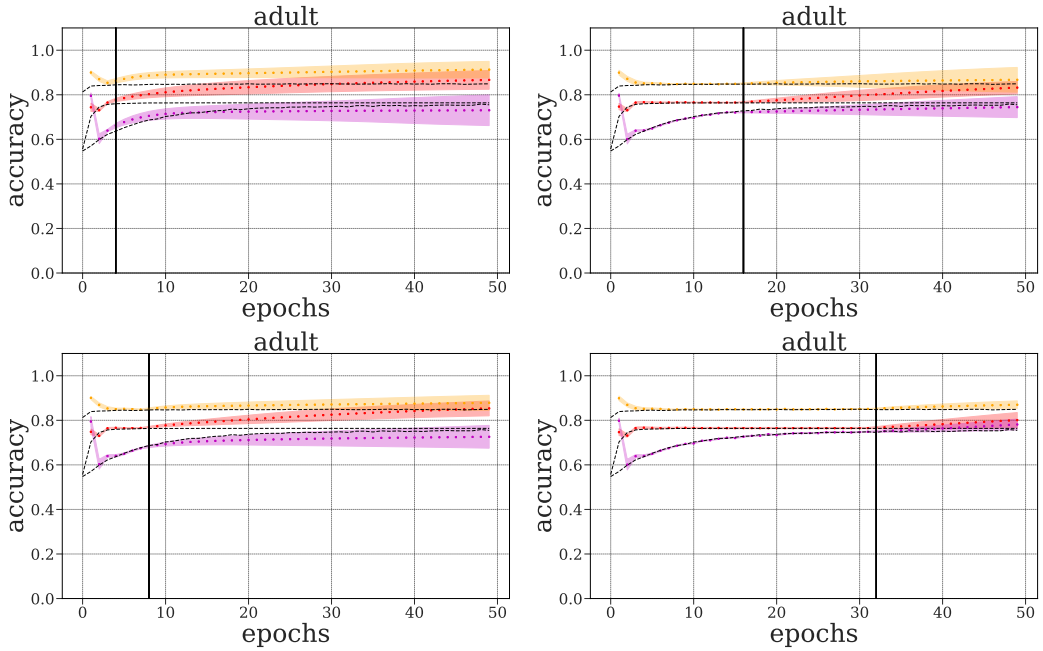


Figure 27: Qualitative assessment of VRNN predictions on the Adult benchmark when trained on MNIST for different numbers of observed epochs at test time (the black vertical line).

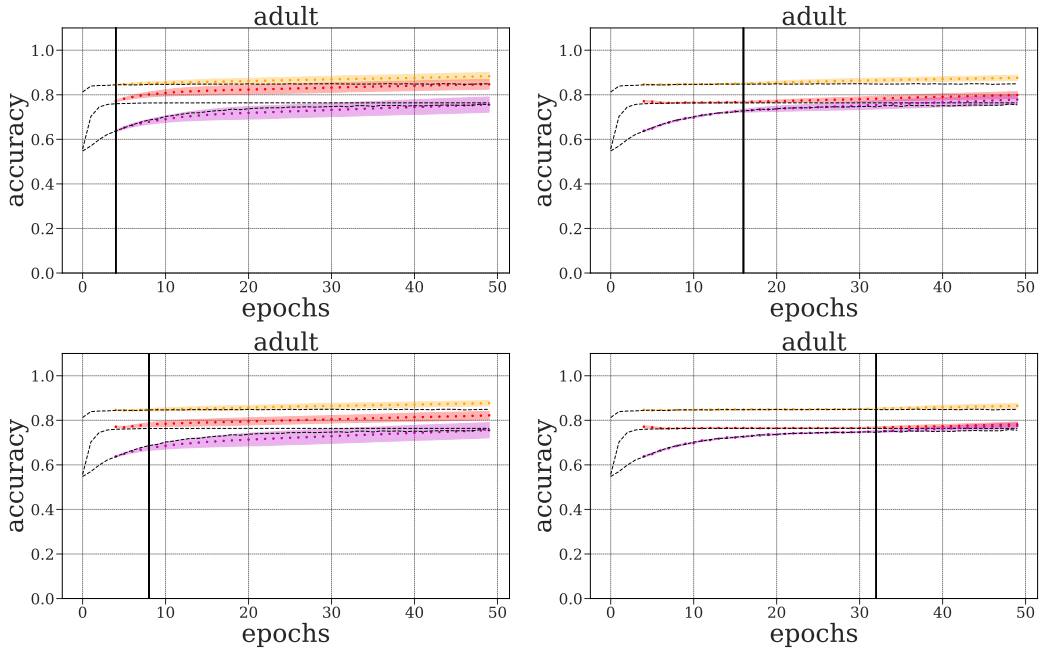


Figure 28: Qualitative assessment of RF 4 predictions on the Adult benchmark when trained on MNIST for different numbers of observed epochs at test time (the black vertical line).

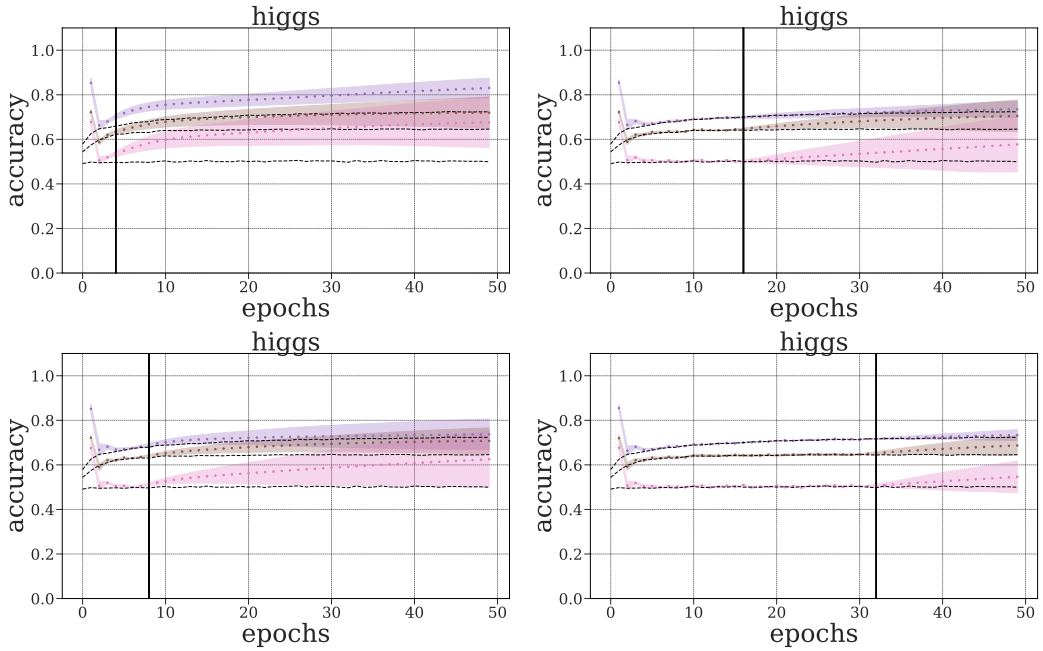


Figure 29: Qualitative assessment of VRNN predictions on the Higgs benchmark when trained on MNIST for different numbers of observed epochs at test time (the black vertical line).

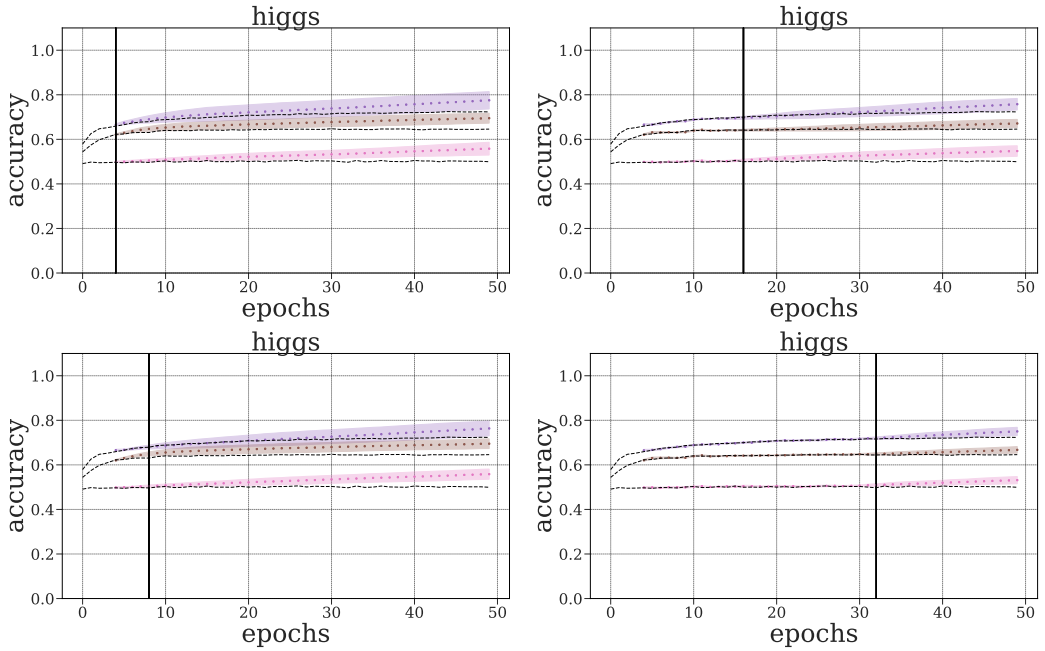
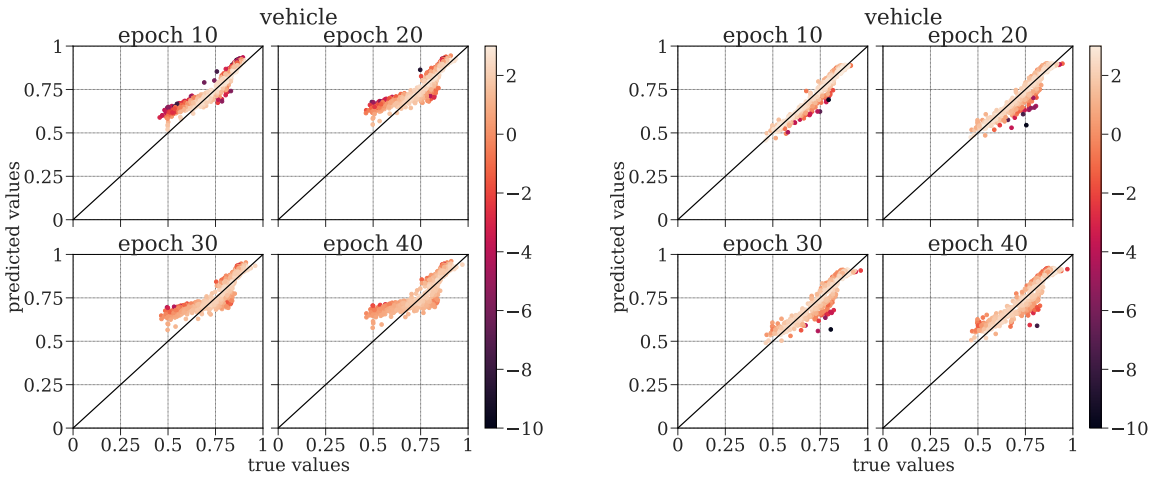


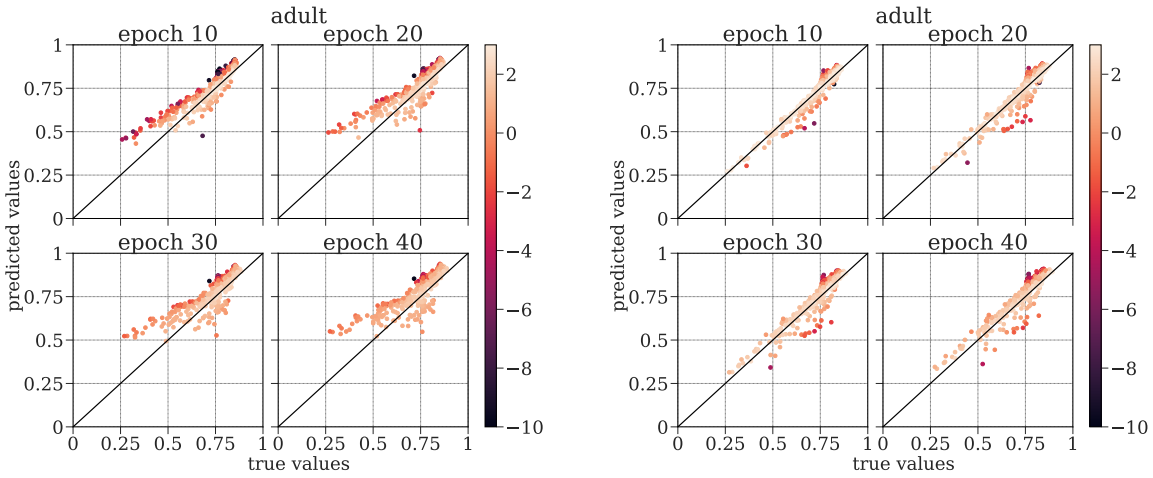
Figure 30: Qualitative assessment of RF 4 predictions on the Higgs benchmark when trained on MNIST for different numbers of observed epochs at test time (the black vertical line).



(a) VRNN

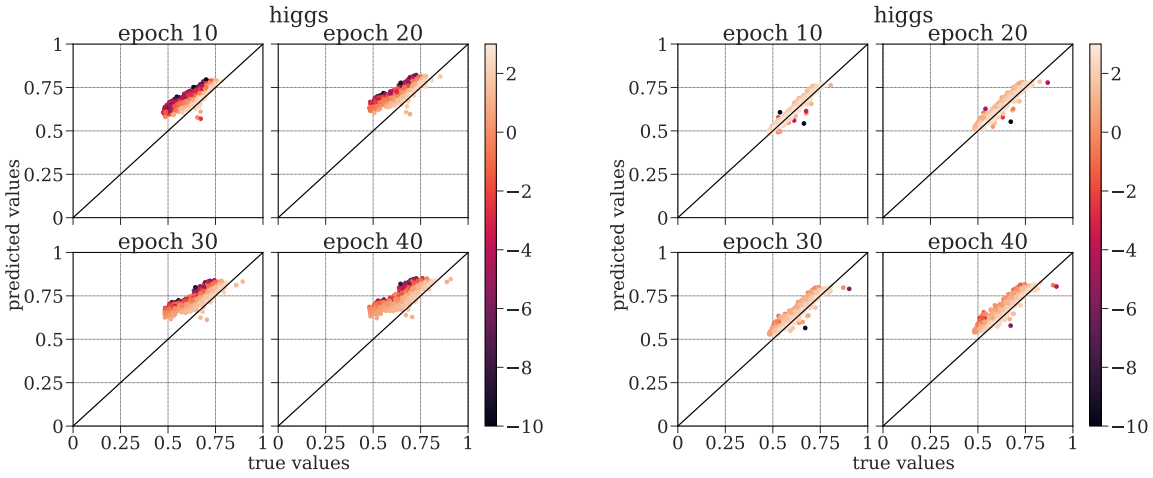
(b) RF 4

Figure 31: The panels show on the horizontal axis the true values and on the vertical axis the predicted values on Vehicle benchmark for VRNN (left) and RF 4 (right) when trained on MNIST. Each point is colored based on its log-likelihood value.



(a) VRNN (b) RF 4

Figure 32: The panels show on the horizontal axis the true values and on the vertical axis the predicted values on the Adult benchmark for VRNN with 4 observed points (left) and RF 4 (right) when trained on MNIST. Each point is colored based on its log-likelihood value.



(a) VRNN (b) RF 4

Figure 33: The panels show on the horizontal axis the true values and on the vertical axis the predicted values on the Higgs benchmark for VRNN with 4 observed points (left) and RF 4 (right) when trained on MNIST. Each point is colored based on its log-likelihood value.