

# Depth Uncertainty Networks for Active Learning



**Chelsea Murray**

Department of Engineering  
University of Cambridge

This dissertation is submitted for the degree of  
*Master of Philosophy in Machine Learning and Machine Intelligence*

Corpus Christi College

August 2021



## Declaration

I, Chelsea Murray of Corpus Christi College, being a candidate for the MPhil in Machine Learning and Machine Intelligence, hereby declare that this report and the work described in it are my own work, unaided except as may be specified below, and that the report does not contain material that has already been used to any substantial extent for a comparable purpose.

*Existing software:* The software used in this thesis extends upon code contained in the following repository by Antorán et al. (2020): <https://github.com/cambridge-mlg/DUN>. All software is implemented in Python using PyTorch. No proprietary software tools were used.

Word count: 14,964

Chelsea Murray  
August 2021



## **Acknowledgements**

I would like to extend my gratitude first and foremost to my supervisors, James Allingham and Javier Antorán, for being incredibly generous with their time and ideas throughout the course of this project. Their enthusiasm and support have made the project a pleasure to work on. I am grateful also to José Miguel Hernández-Lobato for his guidance during the project.

I would like to acknowledge the use of CPU and GPU credits on the University of Cambridge HPC provided courtesy of the Functional Uncertainty in Neural Networks through Stochastic Depth project from the Department of Engineering.

Thank you also to Sebastian Farquhar for helpful clarification about the implementation of the active learning bias experiments.



## Abstract

In many important applications of machine learning, labelled data are often scarce and expensive to obtain. A challenge in such settings is data efficient learning, such that vast quantities of labelled data are not required to attain good model performance. Active learning is one approach to improving data efficiency. Given a fixed labelling budget, the objective of active learning is to identify examples that maximise the expected gain in model performance. This thesis investigates the application of depth uncertainty networks (DUNs), a variant of Bayesian neural networks, to active learning problems. In DUNs, probabilistic inference is performed over the depth of the network, and uncertainty about the optimal depth is translated into model uncertainty via marginalisation.

This thesis proposes and tests several hypotheses about the performance of DUNs in active learning settings. Firstly, the ability to infer depth is expected to enable DUNs to adapt model complexity to the varying size of the training dataset as additional labels are acquired. Experimental evidence validates this hypothesis and shows that this property yields performance advantages. Secondly, we propose that flexibility over depth results in reduced overfitting bias in DUNs, an intuition that is confirmed by empirical results. This finding implies, in theory, that eliminating the bias induced by active sampling of training data should improve the performance of DUNs, but this is found not to be the case in practice. We also investigate whether the prior over depth can be exploited as a further regularisation mechanism, but find that it has minimal impact on the posterior.

Finally, two modifications to existing acquisition strategies are introduced. The first method aims to ameliorate the impact of model misspecification bias, which causes uncertainty estimates of DUNs and other complex models to be unreasonably large outside the range of observed data. The second method addresses the issue of correlation in batch-mode acquisition by targeting the diversity of the acquired batch.





# Table of contents

<b>List of figures</b>	<b>xi</b>
<b>List of tables</b>	<b>xv</b>
<b>Nomenclature</b>	<b>xvii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Motivation . . . . .	1
1.2 Contributions . . . . .	2
1.3 Thesis outline . . . . .	3
<b>2 Background</b>	<b>5</b>
2.1 Active learning . . . . .	5
2.1.1 Acquisition functions . . . . .	6
2.2 Bayesian deep learning . . . . .	10
2.2.1 Variational inference . . . . .	12
2.2.2 Monte Carlo dropout . . . . .	18
2.2.3 Depth uncertainty networks . . . . .	20
2.3 Bias in active learning . . . . .	23
2.3.1 Unbiased risk estimators . . . . .	24
2.3.2 Interaction of biases in active learning . . . . .	25
<b>3 Hypotheses: addressing biases in active learning with DUNs</b>	<b>27</b>
3.1 DUNs for active learning . . . . .	28
3.2 Addressing misspecification bias . . . . .	29
3.2.1 Correlation in batch acquisition . . . . .	30
3.3 Investigating active learning bias . . . . .	30

<b>4</b>	<b>Results</b>	<b>33</b>
4.1	Experimental setup . . . . .	33
4.1.1	Data . . . . .	33
4.1.2	Model specifications . . . . .	35
4.2	Acquisition functions . . . . .	36
4.2.1	Toy regression with naïve batch acquisition . . . . .	37
4.2.2	Truncated BALD . . . . .	37
4.2.3	Stochastic BALD . . . . .	39
4.3	DUN and baseline performance . . . . .	43
4.3.1	Toy datasets . . . . .	43
4.3.2	Tabular regression . . . . .	45
4.3.3	Image classification . . . . .	47
4.4	Bias in active learning . . . . .	48
4.4.1	Quantifying active learning bias . . . . .	49
4.4.2	Training with unbiased risk estimators . . . . .	51
4.4.3	Overfitting bias . . . . .	55
4.4.4	Impact of the depth prior . . . . .	57
<b>5</b>	<b>Conclusion</b>	<b>63</b>
5.1	Summary of findings . . . . .	63
5.2	Limitations and future work . . . . .	64
	<b>References</b>	<b>67</b>
	<b>Appendix A Additional results</b>	<b>73</b>
A.1	Acquisition function comparisons . . . . .	73
A.2	DUN and baselines comparisons . . . . .	77
A.3	Active learning bias experiments . . . . .	79
A.4	Alternative depths for baseline methods . . . . .	81

# List of figures

2.1	Representation of a MAP neural network and a BNN. . . . .	11
2.2	DUN computational model. . . . .	21
2.3	Graphical models of BNNs and DUNs. . . . .	22
2.4	Illustrative linear regression under active learning bias. . . . .	26
4.1	Toy datasets with train-test splits. . . . .	34
4.2	Test NLL vs. number of training points for DUNs evaluated on toy datasets. Maximum entropy and BALD acquisition functions are compared to a random acquisition baseline. . . . .	37
4.3	Example acquisition steps with standard BALD and truncated BALD acquisition functions. . . . .	39
4.4	Initial acquisition steps with truncated BALD acquisition. . . . .	40
4.5	NLL of stochastic BALD with varying temperatures. . . . .	41
4.6	Test NLL vs. number of training points for DUNs evaluated on UCI datasets. Truncated and stochastic BALD acquisition functions are compared to standard BALD. . . . .	42
4.7	Test NLL vs. number of training points for DUNs evaluated on UCI datasets. Maximum entropy and stochastic BALD acquisition functions are compared to a random acquisition baseline. . . . .	43
4.8	Classification accuracy and NLL vs. number of training points for DUNs evaluated on MNIST. Maximum entropy, standard BALD, and stochastic BALD acquisition functions are compared to a random acquisition baseline. . . . .	44
4.9	NLL vs. number of training points for DUNs, MCDO and MFVI evaluated on toy datasets. . . . .	45
4.10	Model fit of a DUN, MCDO and MFVI on the Matern and Wiggle datasets. . . . .	46
4.11	NLL vs. number of training points for DUNs, MCDO and MFVI using 2,000 training epochs. . . . .	46
4.12	Posterior probabilities over depth for DUNs trained on toy datasets. . . . .	47

4.13	NLL vs. number of training points for DUNs, MCDO and MFVI evaluated on UCI datasets. . . . .	48
4.14	Posterior probabilities over depth for DUNs trained on UCI regression datasets. . . . .	49
4.15	Accuracy and NLL vs. number of training points for DUNs and MCDO evaluated on MNIST. . . . .	50
4.16	Active learning bias for DUNs, evaluated using $\tilde{R}$ and $\tilde{R}_{LURE}$ . . . . .	51
4.17	Active learning bias for DUNs and MCDO, evaluated using $\tilde{R}$ and $\tilde{R}_{LURE}$ . . . . .	52
4.18	NLL for DUNs trained with $\tilde{R}$ and $\tilde{R}_{LURE}$ . . . . .	53
4.19	NLL for DUNs and MCDO trained with $\tilde{R}$ and $\tilde{R}_{LURE}$ . . . . .	54
4.20	Overfitting bias for DUNs and MCDO trained with $\tilde{R}$ and $\tilde{R}_{LURE}$ . . . . .	56
4.21	NLL for DUNs with uniform and decaying priors trained with $\tilde{R}$ and $\tilde{R}_{LURE}$ . . . . .	58
4.22	Posterior probabilities for DUNs with uniform and exponentially decaying priors. . . . .	59
4.23	NLL vs. number of training points for DUNs using uniform and exponentially decaying priors. . . . .	60
4.24	Overfitting bias for DUNs with uniform and decaying priors trained with $\tilde{R}$ and $\tilde{R}_{LURE}$ . . . . .	62
A.1	NLL vs. number of training points for DUNs evaluated on toy datasets. Truncated BALD and stochastic BALD acquisition functions are compared to standard BALD. . . . .	73
A.2	NLL vs. number of training points for DUNs evaluated on toy datasets. Stochastic BALD acquisition with and without prior truncation are compared. . . . .	74
A.3	NLL vs. number of training points for DUNs evaluated on toy datasets. Maximum entropy and stochastic BALD are compared to a random acquisition baseline. . . . .	75
A.4	RMSE vs. number of training points for DUNs evaluated on toy datasets. Maximum entropy and stochastic BALD are compared to a random acquisition baseline. . . . .	75
A.5	RMSE vs. number of training points for DUNs evaluated on UCI datasets. Maximum entropy and stochastic BALD acquisition functions are compared to a random acquisition baseline. . . . .	76
A.6	RMSE vs. number of training points for DUNs, MCDO and MFVI evaluated on toy datasets. . . . .	77
A.7	RMSE vs. number of training points for DUNs, MCDO and MFVI evaluated on UCI datasets. . . . .	78
A.8	RMSE for DUNs trained with $\tilde{R}$ and $\tilde{R}_{LURE}$ . . . . .	79

---

A.9	NLL for DUNs with decaying priors trained with $\tilde{R}$ and $\tilde{R}_{LURE}$ . . . . .	80
A.10	NLL vs. number of training points for baseline methods of varying depths. .	83
A.11	RMSE vs. number of training points for baseline methods of varying depths.	85



# List of tables

4.1	Summary of datasets and active learning specifications. . . . .	35
-----	---	----





# Nomenclature

## Acronyms / Abbreviations

BALD	Bayesian active learning by disagreement
BNN	Bayesian neural network
DUN	Depth uncertainty network
ELBO	Evidence lower bound
KL	Kullback-Leibler
LURE	Levelled unbiased risk estimator
MAP	Maximum a-posteriori
MC	Monte Carlo
MCDO	Monte Carlo dropout
MCMC	Markov Chain Monte Carlo
MFVI	Mean-field variational inference
MLE	Maximum likelihood estimation
NLL	Negative log-likelihood
PURE	Plain unbiased risk estimator
RMSE	Root mean-squared error
SGD	Stochastic gradient descent
SGVB	Stochastic gradient variational Bayes
VI	Variational inference



# 1

## Introduction

### 1.1 Motivation

Much of the success of modern machine learning has been facilitated by the availability of vast quantities of data that have been generated in recent decades. Deep learning models, in particular, tend to be characterised by a dependence on large pools of training data in order to generalise well. Krizhevsky et al. (2012), for example, achieved state-of-the-art image classification performance (at the time of publication) by training a deep convolutional neural network on hundreds of gigabytes of labelled images. A second highly publicised example is the Generative Pre-Trained Transformer 3 (GPT-3) language model, which relies on pre-training with an enormous corpus of 45TB of diverse text data in order to learn the ability to generate remarkably human-like written text (Brown et al., 2020).

Despite enabling important advances in the field, such dependence on massive datasets is not necessarily desirable, for both practical and theoretical reasons. From a modelling perspective, it is reasonable to expect that a model should extract as much information as possible from the available data in order to maximise performance on a given task. A reliance on vast quantities of data is therefore potentially a sign of a poorly specified modelling framework that does not learn in a data-efficient manner. A practical consideration is that, in the context of supervised learning, training requires labelled data, which are in many cases scarce or difficult to obtain. Applications of machine learning in fields such as medical imaging or speech recognition, for example, depend on the input of domain experts to annotate images or speech utterances, often a time-consuming and tedious process (Settles, 2010).

Unlabelled data, in contrast, are in many situations abundant and easy to obtain, and we often wish to make use of these data. Active learning is a framework that aims to learn

in such settings, by using measures of model uncertainty to determine which of a set of unlabelled candidate points are likely to be most informative, and obtaining labels only for this subset of the data. The principle behind this approach is that learning can be made more efficient by intelligently selecting examples that provide the most information about the correct values of the model parameters (Cohn et al., 1995). Standard deep learning models do not lend themselves well to the active learning paradigm out-of-the-box, as they are treated as deterministic functions that do not provide measures of model uncertainty alongside their predictions. We consider Bayesian neural networks (BNNs), which cast neural networks as probabilistic models by treating network parameters as random variables.

Standard BNNs have been applied to active learning with empirical success in previous work (Gal et al., 2017; Huang et al., 2018). This thesis focuses instead on a recently proposed form of BNN, depth uncertainty networks (DUNs), in which probabilistic inference is performed over the depth of the network rather than over its weights (Antorán et al., 2020). An interesting feature of DUNs is that predictions from different depths of the network correspond to different types of predictive functions—shallow networks induce simple functions, while deeper networks induce more complex functions. The aim of this thesis is to leverage this property in the context of active learning, and to investigate whether the ability to infer depth can allow DUNs to overcome some of the challenges of active learning with deep neural networks, with a particular focus on biases in active learning.

## 1.2 Contributions

The primary contribution of this thesis is an empirical evaluation of a series of hypotheses about biases in active learning and the performance of DUNs on active learning problems. We propose that **variable-depth networks provide advantages in the active learning setting** by enabling the model to better adapt to the complexity of the dataset, and confirm this intuition by benchmarking DUNs’ performance against that of other BNN methods. A second hypothesis is that **the prior over depth can be manipulated in order to minimise overfitting bias** when there are few labelled data points. In practice, the impact of the prior in DUNs is shown to be negligible. A detailed analysis of different sources of bias in active learning is conducted, showing that **overfitting bias is generally smaller in DUNs** than in other BNN methods. We apply a recently proposed unbiased risk estimator that **corrects for active learning bias**, but find that it does not improve performance for DUNs. Finally, we propose a modification to an information-based acquisition strategy, termed truncated BALD, which **mitigates bias caused by model misspecification**. A further proposed modi-

fication, stochastic BALD, is shown to **improve diversity in batch acquisition** and partially recuperate information loss due to correlated acquisitions.

### 1.3 Thesis outline

The remainder of this report is structured as follows. Chapter 2 introduces active learning and describes the acquisition strategies that are considered in this work. This is followed by a review of BNN inference methods, as well as an explanation of DUNs. We also detail, for each of these methods, how the uncertainty estimates required in active learning are computed. The chapter concludes by reviewing recent work on bias in active learning. Chapter 3 presents the hypotheses we wish to evaluate empirically and explains the methodological approaches adopted for this evaluation. The experimental setup is detailed in chapter 4, followed by an analysis of empirical results. Finally, chapter 5 summarises the findings and conclusions from the thesis and proposes directions for further research.



## 2

# Background

This chapter begins with a review of active learning, and of strategies for active data acquisition, in section 2.1. Section 2.2 introduces important concepts in Bayesian deep learning and describes the BNN methods implemented in this work, including DUNs, in section 2.2.3. In particular, we link the estimation of model predictive uncertainty in each of these methods to the acquisition functions outlined in section 2.1.1. Finally, section 2.3 discusses recent work on bias in active learning and approaches to addressing this bias.

## 2.1 Active learning

In supervised learning settings, the requirement for labelled data can limit the usefulness of available data sources, where labels are costly, expensive, or otherwise difficult to obtain (Jing and Tian, 2020). Given such a setting, the aim of active learning is to produce a model with as high a level of performance as possible using as few labelled data points as possible. In this framework, a learner is initially trained on a small labelled subset of the available data, and additional unlabelled points are selected via an *acquisition function* to be labelled by an external *oracle* (for example, a human expert) (Settles, 2010). The learner is subsequently retrained with the additional labelled points included in the training set, and the process of data acquisition and retraining is repeated iteratively until the targeted performance level is achieved or the query budget is exhausted. By allowing the learner to select candidate points that are expected to be most informative, training ideally becomes more data-efficient, thereby minimising the cost of obtaining a labelled dataset for a given level of model performance.<sup>1</sup>

---

<sup>1</sup>A more accurate formulation of the objective of active learning is to identify the *set* of points (of size equal to the query budget) that are maximally informative (Kirsch et al., 2019). Evaluating all possible subsets of the pool set is, however, intractable, so in practice a greedy process involving iterative acquisition is implemented.

This work concentrates on pool-based active learning, applicable in cases where a large pool of unlabelled data is easy to obtain, but labelling is expensive (Lewis and Gale, 1994). An alternative scenario is online or stream-based active learning, in which a determination about whether or not to request a label for a given data point is made in an online fashion, as single data points are sampled sequentially from the actual distribution (Tong and Koller, 2001). We focus on the former configuration since it is more prevalent than online active learning among application papers, as noted by Settles (2010).

### 2.1.1 Acquisition functions

The key feature distinguishing active learning from *passive learning* is the use of an acquisition function to rank points in the pool set by their potential informativeness. Given a model with parameters  $\theta$  trained on training data  $\mathcal{D}_{\text{train}} = \{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^{N_{\text{train}}}$ , where  $\mathbf{y} \in \mathcal{Y}$ , the acquisition function  $\alpha(\cdot)$  is a function of  $\mathbf{x}$  that scores all unlabelled examples in the pool set  $\mathcal{D}_{\text{pool}} = \{\mathbf{x}_j\}_{j=1}^{N_{\text{pool}}}$ . These scores are used to select the next point  $\mathbf{x}^*$  to be labelled:

$$\mathbf{x}^* = \underset{\mathbf{x} \in \mathcal{D}_{\text{pool}}}{\operatorname{argmax}} \alpha(\mathbf{x}; \theta, \mathcal{D}_{\text{train}}). \quad (2.1)$$

The literature on acquisition strategies in active learning is vast. Following MacKay (1992a), this work considers information theoretic approaches, which seek to choose  $\mathbf{x}^*$  that are most informative about the values of  $\theta$ . In the Bayesian framework, this equates to maximally reducing the uncertainty in the posterior over model parameters. Two quantities commonly used to capture this uncertainty are the predictive entropy and mutual information.

#### Maximum entropy

Shannon's entropy (Shannon, 1948), defined as

$$\mathbb{H}[\mathbf{y} | \mathbf{x}, \mathcal{D}_{\text{train}}] \equiv - \sum_{\mathbf{y} \in \mathcal{Y}} p(\mathbf{y} | \mathbf{x}, \mathcal{D}_{\text{train}}) \log p(\mathbf{y} | \mathbf{x}, \mathcal{D}_{\text{train}}), \quad (2.2)$$

is an information theoretic measure of the amount of uncertainty in the distribution  $p(\mathbf{y} | \mathbf{x}, \mathcal{D}_{\text{train}})$ . Larger values of  $\mathbb{H}[\mathbf{y} | \mathbf{x}, \mathcal{D}_{\text{train}}]$  reflect a distribution that assigns more uniform probabilities across the possible values of  $\mathbf{y}$ , indicating greater uncertainty in the prediction. We can thus use this quantity to define an acquisition function:

$$\alpha_{\text{Entropy}}(\mathbf{x}; \theta, \mathcal{D}_{\text{train}}) = \mathbb{H}[\mathbf{y} | \mathbf{x}, \mathcal{D}_{\text{train}}]. \quad (2.3)$$



Maximising  $\alpha_{\text{Entropy}}(\mathbf{x}; \theta, \mathcal{D}_{\text{train}})$  corresponds to selecting the datum from  $\mathcal{D}_{\text{pool}}$  for which the predictive distribution under the current model contains the least information.

Shannon’s entropy is defined for variables taking discrete values, so is straightforward to apply to the classification case, where the summation in equation (2.2) is computed over the prediction for each category. For regression problems, however, the predictive distribution is a mixture of Gaussians, for which the entropy must be approximated.<sup>2</sup> The approach adopted in this work is to obtain repeated samples of  $\mathbf{y}$  using the methods described in section 2.2 for each BNN. A histogram over the continuous-valued samples can be constructed by allocating the samples to bins, and the entropy over the histogram is calculated. This method is simple to implement and fast to compute, however can be affected by parameter settings such as the number of samples or number of histogram bins used (Depeweg et al., 2018). Alternative approximation methods include nearest neighbour approaches (Kozachenko and Leonenko, 1987) and kernel density estimation (Beirlant et al., 1997). These methods are not implemented as they are not relevant to the main objectives of this work.

### Bayesian active learning by disagreement

MacKay (1992a) proposes a Bayesian approach that aims to maximise the reduction in the model’s posterior entropy after observing the label of a given data point  $\mathbf{x}$ ,

$$\operatorname{argmax}_{\mathbf{x}} \mathbb{H}[\theta \mid \mathcal{D}_{\text{train}}] - \mathbb{E}_{\mathbf{y} \sim p(\mathbf{y} \mid \mathbf{x}, \mathcal{D}_{\text{train}})} [\mathbb{H}[\theta \mid \mathbf{y}, \mathbf{x}, \mathcal{D}_{\text{train}}]]. \quad (2.4)$$

This objective maximises the expected amount of uncertainty in  $\theta$  that is removed by knowing  $\mathbf{y} \mid \mathbf{x}$ . As highlighted by Houlsby et al. (2011), computing the entropy over parameter posteriors is often intractable due to the high dimensionality of the parameter space. This objective can be re-expressed in terms of entropies in output space by observing that the expression inside the  $\operatorname{argmax}$  operator in equation (2.4) is, by definition, the mutual information between the predictions and the model posterior,  $\mathbb{I}[\theta, \mathbf{y} \mid \mathbf{x}, \mathcal{D}_{\text{train}}]$ . Applying the symmetry property of the mutual information and the definition of mutual information, we obtain

$$\mathbb{I}[\theta, \mathbf{y} \mid \mathbf{x}, \mathcal{D}_{\text{train}}] = \mathbb{I}[\mathbf{y}, \theta \mid \mathbf{x}, \mathcal{D}_{\text{train}}] = \mathbb{H}[\mathbf{y} \mid \mathbf{x}, \mathcal{D}_{\text{train}}] - \mathbb{E}_{\theta \sim p(\theta \mid \mathcal{D}_{\text{train}})} [\mathbb{H}[\mathbf{y} \mid \mathbf{x}, \theta]]. \quad (2.5)$$

The acquisition function can hence be written as

$$\alpha_{\text{BALD}}(\mathbf{x}; \theta, \mathcal{D}_{\text{train}}) = \mathbb{H}[\mathbf{y} \mid \mathbf{x}, \mathcal{D}_{\text{train}}] - \mathbb{E}_{\theta \sim p(\theta \mid \mathcal{D}_{\text{train}})} [\mathbb{H}[\mathbf{y} \mid \mathbf{x}, \theta]], \quad (2.6)$$

<sup>2</sup>Refer to section 2.2 for a more detailed treatment of modelling assumptions.

and can be interpreted as selecting the observation that provides the most information about the model parameters. This formulation is termed Bayesian active learning by disagreement (BALD) by Hounsby et al. (2011), as it selects points for which the predictions of individual parameterisations maximally disagree—i.e., where there is **high uncertainty in the predictive posterior on average**, but the **predictions of individual parameter settings are confident**.

Depeweg et al. (2018) provide an alternative interpretation of the BALD acquisition strategy in terms of epistemic and aleatoric uncertainty. Aleatoric uncertainty refers to the component of predictive uncertainty that is driven by inherent noise in the outcome variable, and is not reduced with the observation of more data. Epistemic uncertainty, in contrast, is attributable to a lack of knowledge about the correct model specification, in terms of either the values of model parameters or the model’s functional form (Bhatt et al., 2021). The total uncertainty of the predictive posterior,

$$p(\mathbf{y} \mid \mathbf{x}, \mathcal{D}_{\text{train}}) = \int p(\mathbf{y} \mid \boldsymbol{\theta}, \mathbf{x}) p(\boldsymbol{\theta} \mid \mathcal{D}_{\text{train}}) d\boldsymbol{\theta}, \quad (2.7)$$

can be quantified by the entropy  $\mathbb{H}[\mathbf{y} \mid \mathbf{x}, \mathcal{D}_{\text{train}}]$ . If, instead of integrating over  $\boldsymbol{\theta}$ , we condition on a single setting of the parameters, the quantity  $\mathbb{H}[\mathbf{y} \mid \mathbf{x}, \boldsymbol{\theta}]$  represents the uncertainty in the value of  $\mathbf{y}$  for a given input  $\mathbf{x}$  and parameter configuration. The **expectation of  $\mathbb{H}[\mathbf{y} \mid \mathbf{x}, \boldsymbol{\theta}]$  under  $p(\boldsymbol{\theta} \mid \mathcal{D}_{\text{train}})$**  measures the overall uncertainty in  $\mathbf{y}$  due to observation noise, or aleatoric uncertainty. Subtracting this quantity from the total predictive uncertainty thus leads to the interpretation of the BALD acquisition function as an estimator of the epistemic uncertainty for each candidate in  $\mathcal{D}_{\text{pool}}$ . That is, under BALD we acquire points in input regions of low observation noise—since there is little information to be gained about the optimal model form in these regions—but high model uncertainty.

An equivalent interpretation can also be derived by considering the predictive variance, rather than the entropy, as a measure of uncertainty (Depeweg et al., 2018). Under the assumption of Gaussian likelihood in regression settings, the predictive entropy is a monotonic function of the variance (Settles, 2010).<sup>3</sup> As such,  $\alpha_{\text{BALD}}$  is computed using predictive variance in place of entropy for regression problems, in order to avoid the approximations required when computing the entropy of a continuous distribution.

For the general case, computing the quantities comprising  $\alpha_{\text{Entropy}}$  and  $\alpha_{\text{BALD}}$  is intractable (regardless of whether variance or entropy is the uncertainty measure of interest). In sec-

<sup>3</sup>Refer to section 2.2 for a more detailed treatment of modelling assumptions and uncertainty quantification in the models under consideration.

tion 2.2 we detail how these functions are computed in practice for each of the model classes considered in this work.

### Random acquisition

Random acquisition is used as a baseline to assess the effectiveness of passive learning strategies. Candidates are sampled randomly from  $\mathcal{D}_{\text{pool}}$  with uniform probability:

$$\alpha_{\text{Random}}(\mathbf{x}; \theta, \mathcal{D}_{\text{train}}) = u, u \sim \mathcal{U}(0, 1). \quad (2.8)$$

### Batch-aware methods

In the classical setting, active learning algorithms select a single point to be added to  $\mathcal{D}_{\text{train}}$  at each acquisition step (*serial* acquisition); in many cases, however, this is an impractical approach. Where labelling is performed by a human annotator, such as a medical expert, it is a much more efficient use of the expert’s resources to request annotations for a batch of examples at once, rather than repeatedly requesting labels for single examples. Serial acquisition is also generally unsuitable when using overparameterised models such as neural networks, as a single example is likely to have a negligible impact on the learning regime (Sener and Savarese, 2018). In addition, although the primary concern in active learning is to minimise the cost of labelling, model retraining is typically also expensive, so reducing the frequency of retraining to a batch level rather than after each acquired example is desirable.

Batch acquisition is, however, a more challenging problem than acquiring individual examples. Naïvely applying the greedy algorithms presented above, by selecting the  $b$ -best examples ranked by these acquisition functions, tends to produce correlated queries (Sener and Savarese, 2018). Although individually informative, similar points are likely to contain overlapping information, rendering labelling efforts partially redundant (Settles, 2011).

Several batch-aware active learning algorithms have been proposed that aim to take into account the diversity of candidate points in the selection criterion. BatchBALD, for example, extends  $\alpha_{\text{BALD}}$  to measure the mutual information between a joint of  $b$  data points and the model parameters:

$$\alpha_{\text{BatchBALD}}(\{\mathbf{x}_1, \dots, \mathbf{x}_b\}; \theta, \mathcal{D}_{\text{train}}) = \mathbb{I}[\mathbf{y}_1, \dots, \mathbf{y}_b, \theta \mid \mathbf{x}_1, \dots, \mathbf{x}_b, \mathcal{D}_{\text{train}}], \quad (2.9)$$

a formulation that considers the dependencies within an acquisition batch and thus identifies *batches* of informative points (Kirsch et al., 2019). An alternative approach is to focus purely

on sampling for diversity, as in Sener and Savarese (2018), who propose framing active learning as a core-set selection problem. This approach equates to identifying unlabelled points that are not well represented in the current  $\mathcal{D}_{\text{train}}$ . Both of these methods are intractable and require greedy approximations in their implementation. Other recent work on batch-mode active learning focussing on diversity sampling include Pinsler et al. (2019) and Ash et al. (2020), among others.

The experiments discussed in chapter 4 of this work generally implement batch acquisition rather than serial acquisition due to the reasons outlined at the beginning of this section. Given that the problem of batch-mode active learning is already investigated in the works referenced above and by others, this thesis does not provide an in-depth treatment of the topic and does not implement the batch-aware methods described here. We instead propose and evaluate an ad-hoc technique for addressing correlation in batch queries in section 4.2.3, whose purpose is to ensure that information-based acquisition performs at least as well as random, and therefore that active learning performance reflects the quality of model uncertainty estimates. The batch-mode results reported in chapter 4 can therefore be interpreted as a lower bound on the performance of the models analysed.

## 2.2 Bayesian deep learning

Applying deep learning in an active learning framework poses several challenges. Perhaps most crucially, many active learning acquisition functions rely on measures of model uncertainty, which are generally not provided by standard neural networks. Second, active learning aims to attain good generalisation performance using as small a training set as possible. Deep neural networks, on the other hand, are prone to overfitting and thus tend to generalise poorly in low data regimes. Finally, training deep models can be difficult and computationally expensive, however training is a process that must be repeated several times during the course of active learning.

Bayesian deep learning provides a solution to the first two of these issues. In BNNs, weights are treated as random variables and are represented by probability distributions, rather than by point estimates, as illustrated in figure 2.1, and in figure 2.3 as a graphical model. A prior distribution  $p(\theta)$  over the weights is defined, and the posterior is obtained by applying Bayes' rule, as follows:

$$p(\theta|\mathcal{D}_{\text{train}}) = \frac{p(\mathcal{D}_{\text{train}}|\theta)p(\theta)}{p(\mathcal{D}_{\text{train}})}. \quad (2.10)$$

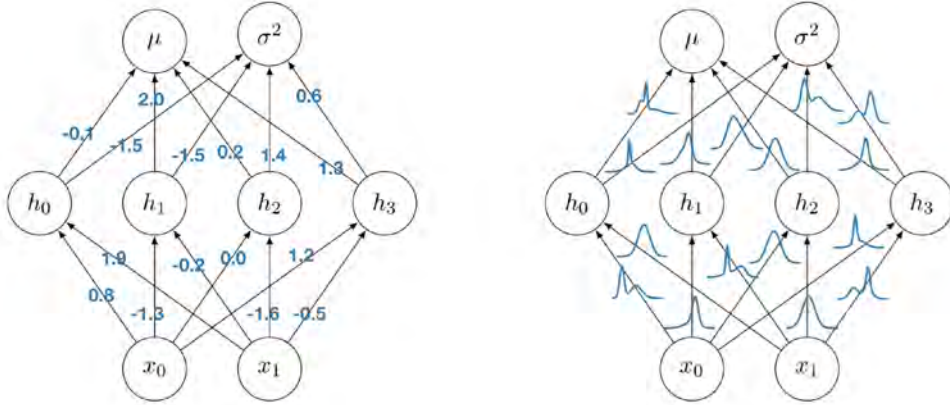


Figure 2.1: Left: In a standard neural network each weight has a maximum likelihood point estimate value. Right: In a BNN each weight is represented by a distribution (Antorán, 2019).

Marginalisation of the posterior distribution over weights yields the predictive distribution, given by equation (2.13), which can be used to obtain the uncertainty estimates required in active learning. By accounting for uncertainty in the weight values, BNNs are more robust to overfitting than deterministic networks, especially for smaller training sets (MacKay, 1992b; Neal, 1995).

To introduce the theory behind BNNs, we consider a neural network as a probabilistic model  $p(\mathbf{y} | \mathbf{x}, \theta)$  that maps an input  $\mathbf{x}$  to probabilities over the space of possible outputs  $\mathbf{y} \in \mathcal{Y}$  using the set of parameters  $\theta$ . A standard approach to learning the parameters of this model uses maximum likelihood estimation (MLE) to generate point estimates  $\theta^{MLE}$  of the weights, being the values that maximise the log-likelihood of the observed training data  $\mathcal{D}_{\text{train}}$ :

$$\theta^{MLE} = \underset{\theta}{\operatorname{argmax}} \log p(\mathcal{D}_{\text{train}} | \theta). \quad (2.11)$$

It is a well-known phenomenon that using MLE with flexible models such as neural networks typically leads to overfitting, as the only incentive during training is to fit the training data as well as possible (Bishop, 2007). One approach to regularising MLE is maximum a-posteriori (MAP) estimation. A prior  $p(\theta)$  is placed on the weights and point estimates are obtained via

$$\begin{aligned} \theta^{MAP} &= \underset{\theta}{\operatorname{argmax}} \log p(\theta | \mathcal{D}_{\text{train}}) \\ &= \underset{\theta}{\operatorname{argmax}} \log p(\mathcal{D}_{\text{train}} | \theta) p(\theta) \\ &= \underset{\theta}{\operatorname{argmax}} \log p(\mathcal{D}_{\text{train}} | \theta) + \log p(\theta), \end{aligned} \quad (2.12)$$

which embodies the assumption that the posterior  $p(\theta \mid \mathcal{D}_{\text{train}})$  is well-approximated by a point mass at its mode. Although more robust to overfitting than MLE, MAP estimation still relies on point estimates of the parameters and therefore fails to quantify confidence in the resulting predictions.

BNNs, in contrast, estimate the posterior probability distribution of the weights, given by equation (2.10). The predictive distribution is obtained by averaging over all possible settings of the weights, using the posterior  $p(\theta \mid \mathcal{D}_{\text{train}})$  to weight the predictions made under each of these settings:

$$p(\mathbf{y}_* \mid \mathbf{x}_*, \mathcal{D}_{\text{train}}) = \mathbb{E}_{p(\theta \mid \mathcal{D}_{\text{train}})}[p(\mathbf{y}_* \mid \mathbf{x}_*, \theta)]. \quad (2.13)$$

This approach is equivalent to performing Bayesian model averaging over an infinite number of ensemble elements, where the variability across predictions of these ensembled networks can be interpreted as a measure of the model’s predictive uncertainty (Blundell et al., 2015).

In general, exact Bayesian inference over the weight-space of a neural network is intractable, necessitating the use of approximations. Laplace’s approximation (MacKay, 1992b) and Markov Chain Monte Carlo (MCMC) methods (Neal, 1995) are two of the earliest proposed approaches. This thesis considers variational inference (VI)-based methods (Graves, 2011; Hinton and Van Camp, 1993), including mean-field variational inference (MFVI) (Blundell et al., 2015), reviewed in section 2.2.1, and Monte Carlo dropout (MCDO) (Gal and Ghahramani, 2016), described in section 2.2.2.

### 2.2.1 Variational inference

VI methods attempt to solve the problem of estimating the intractable posterior by proposing a family of tractable distributions  $\mathcal{Q}$ , and finding the distribution  $q$  within that family that best approximates the true posterior (Blei et al., 2017; Wainwright and Jordan, 2008). This is done by learning the parameters  $\phi$  of the variational distribution  $q(\theta \mid \phi)$  such that the Kullback-Leibler (KL) divergence (Kullback and Leibler, 1951) between  $q(\theta \mid \phi)$  and the

true posterior  $p(\theta \mid \mathcal{D}_{\text{train}})$  is minimised:

$$\begin{aligned}
\phi^* &= \underset{\phi}{\operatorname{argmin}} \operatorname{KL}[q(\theta \mid \phi) \parallel p(\theta \mid \mathcal{D}_{\text{train}})] \\
&= \underset{\phi}{\operatorname{argmin}} \int q(\theta \mid \phi) \log \frac{q(\theta \mid \phi)}{p(\theta \mid \mathcal{D}_{\text{train}})} d\theta \\
&= \underset{\phi}{\operatorname{argmin}} \int q(\theta \mid \phi) \log \frac{q(\theta \mid \phi)}{p(\theta)p(\mathcal{D}_{\text{train}} \mid \theta)} d\theta + \text{const.} \\
&= \underset{\phi}{\operatorname{argmin}} \operatorname{KL}[q(\theta \mid \phi) \parallel p(\theta)] - \mathbb{E}_{q(\theta \mid \phi)}[\log p(\mathcal{D}_{\text{train}} \mid \theta)].
\end{aligned} \tag{2.14}$$

This expression allows the KL divergence to be minimised whilst not containing the explicit form of the posterior  $p(\theta \mid \mathcal{D}_{\text{train}})$ . This transforms the integration problem into one of optimisation, which scales more effectively to large datasets and models (Blei et al., 2017). Once  $q(\theta \mid \phi)$  is selected, prediction proceeds by taking the expectation under the variational posterior rather than the exact posterior,

$$\begin{aligned}
p(\mathbf{y}_* \mid \mathbf{x}_*, \mathcal{D}_{\text{train}}) &= \mathbb{E}_{p(\theta \mid \mathcal{D}_{\text{train}})} [p(\mathbf{y}_* \mid \mathbf{x}_*, \theta)] \\
&\approx \mathbb{E}_{q(\theta \mid \phi)} [p(\mathbf{y}_* \mid \mathbf{x}_*, \theta)].
\end{aligned} \tag{2.15}$$

The objective in equation (2.14) corresponds to a loss function that is commonly referred to as the negative evidence lower bound (ELBO) or variational free energy, denoted:

$$\mathcal{F}(\mathcal{D}_{\text{train}}, \phi) = \operatorname{KL}[q(\theta \mid \phi) \parallel p(\theta)] - \mathbb{E}_{q(\theta \mid \phi)}[\log p(\mathcal{D}_{\text{train}} \mid \theta)]. \tag{2.16}$$

This objective represents a trade-off: **the likelihood cost encourages a close fit to the data**, while **the KL divergence term** imposes a penalty if  $q(\theta \mid \phi)$  moves far from the simple prior distribution, and is thus known as the complexity cost (Blundell et al., 2015).

### Reparameterisation trick and Monte Carlo approximation

A challenge in applying gradient descent to  $\mathcal{F}(\mathcal{D}_{\text{train}}, \phi)$  is computing the derivative of the expectation term. Blundell et al. (2015) apply the reparameterisation trick of Kingma et al. (2015) to this problem as follows: introduce an auxiliary variable  $\varepsilon$  such that  $\theta$  takes the form  $\theta = t(\phi, \varepsilon)$ , where  $t(\phi, \varepsilon)$  is a deterministic function and  $\varepsilon \sim q(\varepsilon)$ . Then for a function

$f$  with derivatives in  $\theta$ :

$$\begin{aligned} \frac{\partial}{\partial \phi} \mathbb{E}_{q(\theta|\phi)}[f(\theta, \phi)] &= \frac{\partial}{\partial \phi} \int f(\theta, \phi) q(\theta | \phi) d\theta \\ &= \frac{\partial}{\partial \phi} \int f(\theta, \phi) q(\varepsilon) d\varepsilon \\ &= \mathbb{E}_{q(\varepsilon)} \left[ \frac{\partial f(\theta, \phi)}{\partial \theta} \frac{\partial \theta}{\partial \phi} + \frac{\partial f(\theta, \phi)}{\partial \phi} \right]. \end{aligned} \quad (2.17)$$

That is, the derivative of the expectation of  $f(\theta, \phi)$  over  $q(\theta | \phi)$  can be re-written as the expectation over  $q(\varepsilon)$  of the derivative of  $f$ . This formulation is known as a pathwise gradient estimator (Mohamed et al., 2020). The random component is isolated into the auxiliary variable  $\varepsilon$ , allowing samples from the (ideally easy-to-sample) distribution  $q(\varepsilon)$  to be transformed into samples from the variational posterior  $q(\theta | \phi)$  via a deterministic function.

Re-writing the objective as

$$\begin{aligned} \mathcal{F}(\mathcal{D}_{\text{train}}, \phi) &= \int q(\theta | \phi) \log \frac{q(\theta | \phi)}{p(\theta)p(\mathcal{D}_{\text{train}} | \theta)} d\theta \\ &= \mathbb{E}_{q(\theta|\varepsilon)} [\log q(\theta | \phi) - \log p(\theta)p(\mathcal{D}_{\text{train}} | \theta)] \end{aligned} \quad (2.18)$$

and letting  $f(\theta, \phi) = \log q(\theta | \phi) - \log p(\theta)p(\mathcal{D}_{\text{train}} | \theta)$  allows us to apply equation (2.17) to the optimisation of equation (2.16). The exact cost is approximated by the MC estimate

$$\begin{aligned} \mathcal{F}(\mathcal{D}_{\text{train}}, \phi) &\approx \sum_{i=1}^n \log q(\theta^{(i)} | \phi) - \log p(\theta^{(i)}) - \log p(\mathcal{D}_{\text{train}} | \theta^{(i)}), \\ \theta^{(i)} &\sim q(\theta | \phi). \end{aligned} \quad (2.19)$$

Optimising equation (2.14) results in an estimation process called ‘‘Bayes by Backprop’’ described by Blundell et al. (2015).

The objective function  $\mathcal{F}(\mathcal{D}_{\text{train}}, \phi)$  is compatible with minibatch optimisation, in which  $\mathcal{D}_{\text{train}}$  is randomly split into equally-sized minibatches of size  $B$  in each epoch. The batch-level objective function is then

$$\mathcal{F}_i(\mathcal{D}_i, \phi) = \frac{B}{N_{\text{train}}} \text{KL}[q(\theta | \phi) || p(\theta)] - \mathbb{E}_{q(\theta|\phi)} [\log p(\mathcal{D}_i | \theta)], \quad (2.20)$$

where  $\mathcal{D}_i$  is the  $i^{\text{th}}$  minibatch (Graves, 2011). During backpropagation, each gradient is averaged over all elements in the minibatch. This procedure of minibatch-based optimisation



of the variational lower bound is known as stochastic gradient variational Bayes (SGVB). The estimator is unbiased and differentiable with respect to  $\phi$ , meaning that the gradient is also unbiased (Kingma and Welling, 2014).

### Local reparameterisation

Although the SGVB estimator is unbiased, the gradient signal can be subject to high variance, potentially resulting in slow convergence (Robbins and Monro, 1951). Kingma et al. (2015) introduce the local reparameterisation trick, which yields a lower-variance and computationally efficient estimator by sampling activations rather than weights. Consider the expected log likelihood term in equation (2.16), which can be re-written as

$$L_{\mathcal{D}_{\text{train}}}(\phi) \approx L_{\mathcal{D}_{\text{train}}}^{\text{SGVB}}(\phi) = \frac{N_{\text{train}}}{B} \sum_{i=1}^B \log p(\mathbf{y}_i | \mathbf{x}_i, \theta = t(\phi, \varepsilon)).^4 \quad (2.21)$$

Letting  $L_i = \log p(\mathbf{y}_i | \mathbf{x}_i, \theta)$  be the contribution of the  $i^{\text{th}}$  observation to the likelihood, the variance of  $L_{\mathcal{D}_{\text{train}}}^{\text{SGVB}}(\phi)$  is given by

$$\text{Var}[L_{\mathcal{D}_{\text{train}}}^{\text{SGVB}}(\phi)] = N_{\text{train}}^2 \left( \frac{1}{B} \text{Var}[L_i] + \frac{B-1}{B} \text{Cov}[L_i, L_j] \right). \quad (2.22)$$

This expression shows that, while the individual variance term scales inversely to  $B$ , the covariance term does not, and hence dominates the variance as  $B$  increases. Obtaining a low-variance estimator can thus be reduced to finding an estimator with  $\text{Cov}[L_i, L_j] = 0$ .

Given that the weights  $\theta$  directly influence the activations, it is possible to exchange sampling weights with sampling the activations. To illustrate how this reduces variance, suppose that the variational posterior is a fully factorised Gaussian,  $q_\phi(\theta_{i,j}) = \mathcal{N}(\mu_{i,j}, \sigma_{i,j})$ , implying that the activations  $b_{m,j}$  are also drawn from a factorised Gaussian:  $q_\phi(b_{m,j} | A) = \mathcal{N}(\gamma_{m,j}, \delta_{m,j})$ , where  $A$  is the input matrix to the activation function with size  $M \times Q$ . Applying the reparameterisation trick described in section 2.2.1 to  $b_{m,j}$ , samples of the activations can be obtained directly from the implied Gaussian as follows:

$$b_{m,j} = \gamma_{m,j} + \sqrt{\delta_{m,j}} \zeta_{m,j}; \quad \zeta_{m,j} \sim \mathcal{N}(0, 1). \quad (2.23)$$

<sup>4</sup>The following analysis extends straightforwardly to the KL term.

Since a separate  $\zeta_{m,j}$  is sampled for each activation, we have that  $\text{Cov}[L_i, L_j] = 0$ , as required. Since the activations are lower-dimensional than the weights, fewer sampling operations are required and the locally reparameterised estimator is also more computationally efficient.

### Variational posterior specification

A common choice for the form of the variational posterior is a fully factorised or mean-field Gaussian distribution, in which case the variational parameters  $\phi$  are a vector of means and variances (Foong et al., 2020). The resulting approximate estimation procedure is referred to as mean-field variational inference (MFVI). The reparameterisation trick implies that sampling  $\theta$  from this posterior can be achieved by transforming samples from a unit Gaussian as follows:

$$\mathbf{w} = t(\phi, \varepsilon) = \mu + \sigma \circ \varepsilon; \quad \varepsilon \sim \mathcal{N}(0, I), \quad (2.24)$$

where  $\mu$  and  $\sigma$  are the variational posterior mean and standard deviation. This allows path-based derivatives to be calculated with respect to the variational parameters  $\phi$  and standard stochastic backpropagation techniques to be used.

The choice of mean-field Gaussians as the family of approximating distributions is motivated primarily by considerations of computational convenience. Such distributions are largely free of numerical issues, and provide each weight with its own measure of uncertainty whilst only doubling the number of parameters to be optimised ( $(\mu, \sigma)$  parameters per weight) relative to an equivalently-sized regular neural network (Blundell et al., 2015). Modelling the full (as opposed to factorised) covariance, in contrast, would require estimating  $N(N+1)/2$  parameters for a network with  $N$  weights. The mean-field approximation has, however, been shown to have limited expressiveness in function space (Foong et al., 2020). For example, the MFVI estimator tends to underestimate “in-between” uncertainty—uncertainty in regions of low information that lie between regions of high certainty (Foong et al., 2019).

### Computing MFVI uncertainties

We return now to the problem of computing  $\alpha_{\text{Entropy}}$  and  $\alpha_{\text{BALD}}$  (section 2.1.1) for the MFVI estimator. Let  $f(\mathbf{x}_*, \theta)$  be the output of the network for a given test point  $\mathbf{x}_*$  and given weight configuration sampled from the variational posterior,  $\theta \sim q(\theta | \phi)$ . Since the expectation under the variational posterior in equation (2.15) cannot be computed exactly, it

is approximated using  $M$  MC samples:

$$\begin{aligned} p(\mathbf{y}_* | \mathbf{x}_*, \mathcal{D}_{\text{train}}) &= \mathbb{E}_{p(\theta | \mathcal{D}_{\text{train}})} [p(\mathbf{y}_* | \mathbf{x}_*, \theta)] \\ &\approx \frac{1}{M} \sum_{i=1}^M f(\mathbf{x}_*, \theta^{(i)}) ; \quad \theta^{(i)} \sim q(\theta | \phi). \end{aligned} \quad (2.25)$$

For classification models with categorical likelihood function  $p(\mathbf{y} | \mathbf{x}, \theta) = \text{Cat}(\mathbf{y}; f(\mathbf{x}, \theta))$ , the approximate predictive distribution is also categorical. Its entropy can thus be computed exactly by summing over the classes  $\{c_k\}_{k=1}^K$ , yielding  $\alpha_{\text{Entropy}}$ :

$$\begin{aligned} \alpha_{\text{Entropy}}(\mathbf{x}_*; \theta, \mathcal{D}_{\text{train}}) &= \mathbb{H}[\mathbf{y} | \mathbf{x}, \mathcal{D}_{\text{train}}] \\ &= - \sum_{k=1}^K p(\mathbf{y}_* = c_k | \mathbf{x}_*, \mathcal{D}_{\text{train}}) \log p(\mathbf{y}_* = c_k | \mathbf{x}_*, \mathcal{D}_{\text{train}}). \end{aligned} \quad (2.26)$$

For regression models, the presence of a mixture of Gaussians predictive distribution necessitates the approximation of  $\alpha_{\text{Entropy}}(\mathbf{x}_*; \theta, \mathcal{D}_{\text{train}})$ . The histogram-based approach described in section 2.1.1 implies sampling several weight configurations  $\theta^{(i)}$  and collecting the resulting outputs  $f(\mathbf{x}_*, \theta^{(i)})$ . A discrete distribution can be constructed from the samples via binning, and the entropy over this distribution calculated.

Computing  $\alpha_{\text{BALD}}$  likewise requires approximations. In the regression case, consider a homoscedastic Gaussian likelihood function with mean parameterised by the neural network output  $f(\mathbf{x}_*, \theta)$ :  $p(\mathbf{y} | \mathbf{x}, \theta) = \mathcal{N}(\mathbf{y}; f(\mathbf{x}_*, \theta), \sigma^2)$  (the variance  $\sigma^2$  is learnt as a standalone parameter). This induces a mixture of Gaussians predictive posterior distribution. Following Lakshminarayanan et al. (2017), we approximate this mixture with a single Gaussian via moment matching,  $p(\mathbf{y}_* | \mathbf{x}_*, \mathcal{D}_{\text{train}}) \approx \mathcal{N}(\mathbf{y}; \mu_a, \sigma_a^2)$ , with the mean estimated using MC sampling:

$$\mu_a \approx \frac{1}{M} \sum_{i=1}^M f(\mathbf{x}_*, \theta^{(i)}) ; \quad \theta^{(i)} \sim q(\theta | \phi). \quad (2.27)$$

The predictive variance is given by the variance of the Gaussian mixture, which we approximate with MC sampling:

$$\sigma_a^2 \approx \frac{1}{M} \sum_{i=1}^M f(\mathbf{x}_*, \theta^{(i)})^2 - \mu_a^2 + \sigma^2 ; \quad \theta^{(i)} \sim q(\theta | \phi). \quad (2.28)$$

This expression is composed of the **epistemic uncertainty** and **aleatoric uncertainty**. Recall from section 2.1.1 that  $\alpha_{\text{BALD}}$  is equivalent to the epistemic component of predictive variance (under a Gaussian likelihood); we can thus use the part of equation (2.28) measuring the

epistemic uncertainty to compute BALD acquisition function scores for regression problems.

For classification problems, the likelihood is not Gaussian, so  $\alpha_{\text{BALD}}$  is obtained by computing entropies rather than variances. From equation (2.6), obtaining  $\alpha_{\text{BALD}}$  requires computing  $\mathbb{H}[\mathbf{y}_* | \mathbf{x}_*, \mathcal{D}_{\text{train}}]$ , the entropy of the model prediction, and  $\mathbb{E}_{\theta \sim p(\theta | \mathcal{D}_{\text{train}})}[\mathbb{H}[\mathbf{y}_* | \mathbf{x}_*, \theta]]$ , the expectation of the predictive entropy over the posterior of the model parameters. We generate  $M$  MC samples  $f(\mathbf{x}_*, \theta^{(i)})$  and compute  $p(\mathbf{y}_* | \mathbf{x}_*, \mathcal{D}_{\text{train}})$  as in equation (2.25), from which the model entropy can be calculated as

$$\mathbb{H}[\mathbf{y}_* | \mathbf{x}_*, \mathcal{D}_{\text{train}}] = \sum_{k=1}^K p(\mathbf{y}_* = c_k | \mathbf{x}_*, \mathcal{D}_{\text{train}}) \log p(\mathbf{y}_* = c_k | \mathbf{x}_*, \mathcal{D}_{\text{train}}). \quad (2.29)$$

To obtain the expectation of entropies, we compute the entropy of each of the MC samples and average them.

## 2.2.2 Monte Carlo dropout

An alternative approximate inference technique is that proposed by Gal and Ghahramani (2016), who re-interpret dropout as approximate variational inference in deep Gaussian processes. Under this method, uncertainty estimates can be obtained in a more computationally efficient manner than with variational inference. Dropout is a stochastic regularisation technique in which an ensemble of subnetworks is trained by randomly setting the non-output nodes of a base network to zero with probability  $p_{\text{drop}}$  with each training example (Hinton et al., 2012; Srivastava et al., 2014). It has been shown to be effective in preventing overfitting and co-adaptation between weights. Gal and Ghahramani (2016) propose applying dropout also at prediction time, yielding the Monte Carlo dropout (MCDO) estimator. They recast this process as an approximate Bayesian inference method by showing that it effectively minimises the KL divergence between an approximating distribution and a deep Gaussian process (marginalised over its covariance function parameters).

Obtaining measures of predictive uncertainty with the MCDO estimator involves performing  $T$  stochastic forward passes for the input location  $\mathbf{x}_*$ , with each prediction  $y_*(\mathbf{x}_*, \theta^{(t)})$  being generated using a different set of network weights  $\theta^{(t)}$  selected by applying dropout as described above. Each weight configuration can be viewed as a MC sample from the approximate posterior distribution of weights,  $q(\theta | \mathcal{D}_{\text{train}})$ , and each prediction  $\mathbf{y}_*(\mathbf{x}_*, \theta^{(t)})$

as a MC sample from the approximate predictive distribution,

$$q(\mathbf{y}_* | \mathbf{x}_*) = \int p(\mathbf{y}_* | \mathbf{x}_*, \boldsymbol{\theta}) q(\boldsymbol{\theta} | \mathcal{D}_{\text{train}}) d\boldsymbol{\theta}. \quad (2.30)$$

The MCDO estimate of the prediction can be recovered using MC integration:

$$\mathbb{E}_{q(\mathbf{y}_* | \mathbf{x}_*)} [\mathbf{y}_*] \approx \frac{1}{T} \sum_{t=1}^T \mathbf{y}_* (\mathbf{x}_*, \boldsymbol{\theta}^{(t)}). \quad (2.31)$$

Model uncertainty, as measured by the predictive variance, is given by

$$\text{Var}_{q_{\phi}[\mathbf{y}_* | \mathbf{x}_*]} (\mathbf{y}_*) \approx \boldsymbol{\tau}^{-1} + \frac{1}{T} \sum_{t=1}^T \mathbf{y}_* (\mathbf{x}_*, \boldsymbol{\theta}^{(t)})^2 - (\mathbb{E}_{q(\mathbf{y}_* | \mathbf{x}_*)} [\mathbf{y}_*])^2, \quad (2.32)$$

the sample variance of the  $T$  forward passes through the network with dropout applied, plus the inverse model precision  $\boldsymbol{\tau}$  of the deep Gaussian process model (full details of the derivation are given in Gal and Ghahramani (2016)).

An advantage of MCDO is that obtaining uncertainty estimates can be approached as training a standard neural network, requiring adjustments only to the prediction procedure in order to generate  $T$  forward passes with dropout for each test point. Relative to MFVI, which requires the estimation of twice as many network parameters, as well as several samples to be drawn per training example to estimate the ELBO, MCDO is computationally more efficient. The method does, however, pose several important drawbacks. MCDO uncertainty estimates are known to not be well-calibrated by default, with the quality of the estimates depending on the dropout rate  $p_{\text{drop}}$  (Gal, 2016). It has also been shown that the epistemic uncertainty estimate produced by the MCDO estimator is independent of the amount of training data and their variance, meaning that parameter choices such as  $p_{\text{drop}}$  must be calibrated to the expected level of uncertainty (Osband, 2016; Verdoja and Kyrki, 2020). This may be particularly problematic for active learning, since the BALD acquisition function is predicated on the epistemic uncertainty estimates being well-calibrated in order to make sensible selections for labelling.

### Computing MCDO uncertainties

Computing the quantities  $\alpha_{\text{Entropy}}$  and  $\alpha_{\text{BALD}}$  for MCDO can be approached in a similar way to the methods described in section 2.2.1 for MFVI. In framing MCDO as an approximate variational inference technique, Gal and Ghahramani (2016) define the variational posterior

$q(\theta \mid \phi)$  as a distribution over matrices  $\mathbf{W}_i$  for each layer  $i$  in the network, whose columns are randomly set to zero:

$$\begin{aligned} \mathbf{W}_i &= \mathbf{M}_i \cdot \text{diag} \left( [z_{i,j}]_{j=1}^{K_i} \right) \\ z_{i,j} &\sim \text{Bernoulli} (1 - p_{\text{drop}_i}) \text{ for } i = 1, \dots, L, j = 1, \dots, K_{i-1}, \end{aligned} \quad (2.33)$$

where  $K_i$  is the number of hidden units in layer  $i$  and  $L$  the number of layers in the network. The matrices  $\mathbf{M}_i$  and probabilities  $p_{\text{drop}_i}$  are the variational parameters. Sampling from this variational posterior corresponds to sampling  $z_{i,j}$  for each hidden unit and computing the network output with the resulting stochastic weight matrices  $\mathbf{W}_i$  (i.e., performing a forward pass with dropout applied). Having defined  $q(\theta \mid \phi)$  and how to sample from it, calculating  $\alpha_{\text{Entropy}}$  and  $\alpha_{\text{BALD}}$  for both regression and classification models proceeds exactly as described in section 2.2.1 for the MFVI estimator.

### 2.2.3 Depth uncertainty networks

BNNs translate uncertainty in weight-space into predictive uncertainty by marginalising out the posterior over weights (equation (2.13)). The intractability of this posterior necessitates the use of approximate inference methods such as MFVI (section 2.2.1) and MCDO (section 2.2.2). DUNs, in contrast, leverage uncertainty about the appropriate *depth* of the network in order to obtain predictive uncertainty estimates. There are two primary advantages of this approach: 1) the posterior over depth is tractable, mitigating the need for limiting approximations, such as those used in MFVI; and 2) due to the sequential structure of feed-forward neural networks, both inference and prediction can be performed with a single forward pass, making DUNs suitable for deployment in low-resource environments (Antorán et al., 2020). Targeting depth as a source of uncertainty does, however, have a drawback in that the prior over depth has a limited influence over the posterior. Thus, although meaningful priors can in theory be placed over depth, in practice the impact of these priors is minimal. This limitation is discussed in more detail in section 4.4.4.

DUNs are composed of subnetworks of increasing depth, with each subnetwork contributing one prediction to the final model—this is comparable to each sampled weight configuration in BNNs contributing a single prediction to the MC estimator of the predictive distribution given in equation (2.25). The computational model for DUNs is shown in figure 2.2. Inputs are passed through an input block,  $f_0(\cdot)$ , and then sequentially through each of  $D$  intermediate blocks  $\{f_i(\cdot)\}_{i=1}^D$ , with the activations of the previous layer acting as the inputs of the current layer. The activations of each of the  $D + 1$  layers are passed through the output

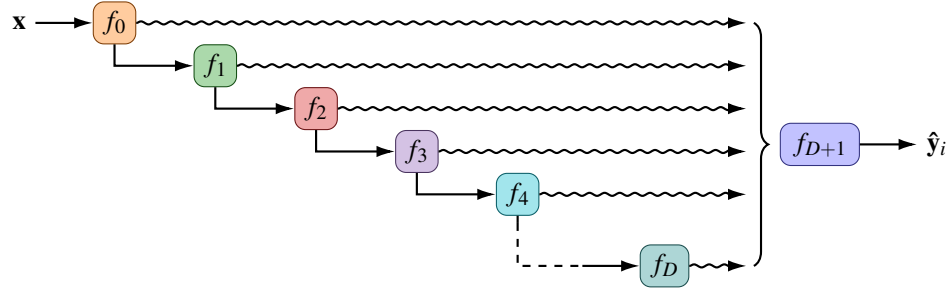


Figure 2.2: DUN computational model (Antorán et al., 2020). Each layer’s activations are passed through the output block, producing per-depth predictions.

block  $f_{D+1}(\cdot)$  to generate per-depth predictions. These are combined via marginalisation over the depth posterior, equivalent to forming an ensemble with networks of increasing depth. Variation between the predictions from each depth can be used to obtain predictive uncertainty estimates, in much the same way that variance in the predictions of different sampled weight configurations yield predictive uncertainties in BNNs.

### Uncertainty over depth

Figure 2.3 compares the graphical model representations of a BNN and a DUN. In BNNs, the weights  $\theta$  are treated as random variables drawn from a distribution  $p_\gamma(\theta)$  with hyperparameters  $\gamma$ . In DUNs, the depth of the network  $d$  is assumed to be stochastic, while the weights are learned as hyperparameters. A categorical prior distribution is assumed for  $d$ , with hyperparameters  $\beta$ :  $p_\beta(d) = \text{Cat}(d | \{\beta_i\}_{i=0}^D)$ . The model’s marginal log likelihood (MLL) is given by

$$\log p(\mathcal{D}_{\text{train}}; \theta) = \log \sum_{i=0}^D \left( p_\beta(d=i) \prod_{n=1}^N p(\mathbf{y}_n | \mathbf{x}_n, d=i; \theta) \right), \quad (2.34)$$

where the likelihood for each depth is parameterised by the corresponding subnetwork’s output:  $p(\mathbf{y} | \mathbf{x}, d=i; \theta) = p(\mathbf{y} | f_{D+1}(\mathbf{a}_i; \theta))$ , where  $\mathbf{a}_i = f_i(\mathbf{a}_{i-1})$ .

The posterior,

$$p(d | \mathcal{D}_{\text{train}}; \theta) = \frac{p(\mathcal{D}_{\text{train}} | d; \theta) p_\beta(d)}{p(\mathcal{D}_{\text{train}}; \theta)}, \quad (2.35)$$

is also a categorical distribution, whose probabilities reflect how well each depth subnetwork explains the data. DUNs leverage two properties of modern neural networks: first, that successive layers have been shown to extract features at increasing levels of abstraction (Zeiler and Fergus, 2014); and second, that networks are typically underspecified, meaning

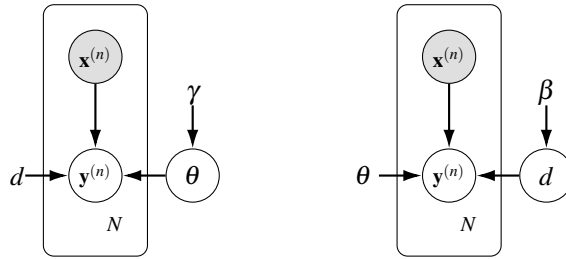


Figure 2.3: Left: graphical model of a BNN. Right: graphical model of a DUN.

that several different models may explain the data well (D’Amour et al., 2020). The first property implies that initial layers in a network specialise on low-level feature extraction, yielding poor predictions from the shallower subnetworks in a DUN. This is handled by the depth posterior assigning low probabilities to earlier layers, in preference for later layers that specialise on prediction. The second property suggests that subnetworks at several depths are able to explain the data simultaneously and thus have non-zero posterior probabilities, yielding the diversity in predictions required to obtain useful estimates of model uncertainty.

### Inference in DUNs

Antorán et al. (2020) find that directly optimising the MLL equation (2.34) with respect to  $\theta$  results in convergence to a local optimum in which all but one layer is attributed a posterior probability of zero. That is, direct optimisation returns a deterministic network of arbitrary depth. The authors instead propose a stochastic gradient VI approach, as described in section 2.2.1 for BNNs, with the aim of separating optimisation of the weights  $\theta$  from the posterior. A surrogate categorical distribution  $q_\phi(d)$  is introduced as the variational posterior. The following ELBO can be derived:

$$\mathcal{L}(\phi, \theta) = \sum_{n=1}^{N_{\text{train}}} \mathbb{E}_{q_\phi(d)} [\log p(\mathbf{y}_n | \mathbf{x}_n, d; \theta)] - \text{KL}(q_\phi(d) || p_\beta(d)), \quad (2.36)$$

allowing  $\theta$  and the variational parameters  $\phi$  to be optimised simultaneously. Antorán et al. (2020) show that under the VI approach, the variational parameters converge more slowly than the weights, enabling the weights to reach a setting at which a positive posterior probability is learnt for several depths.

It is important to note that equation (2.36) can be computed exactly, and that optimising  $L(\phi, \theta)$  is equivalent to exact inference of the true posterior  $p(d | \mathcal{D}_{\text{train}}; \theta)$  in the limit: since both  $q$  and  $p$  are categorical, equation (2.36) is convex and tight at the optimum (i.e.,  $q_\phi(d) = p(d | \mathcal{D}_{\text{train}}; \theta)$ ). Since the expectation term can be computed exactly using the



activations at each layer (recall  $p(\mathbf{y} \mid \mathbf{x}, d = i; \theta) = p(\mathbf{y} \mid f_{D+1}(\mathbf{a}_i; \theta))$ ), the ELBO can be evaluated exactly without MC sampling.

### Computing DUN uncertainties

The predictive posterior is obtained by marginalising out the variational distribution over depth:

$$p(\mathbf{y}_* \mid \mathbf{x}_*, \mathcal{D}_{\text{train}}; \theta) = \sum_{i=0}^D p(\mathbf{y}_* \mid \mathbf{x}_*, d = i; \theta) q_\phi(d = i). \quad (2.37)$$

As in the case of inference, equation (2.37) can be evaluated exactly. As a result, the uncertainty quantities computed in  $\alpha_{\text{Entropy}}$  and  $\alpha_{\text{BALD}}$  are not based on approximations, as required with MFVI and MCDO. For classification problems,  $\alpha_{\text{Entropy}}$  is obtained by computing the entropy of the predicted probability vector.  $\alpha_{\text{BALD}}$  is estimated in the same way as for MFVI and MCDO (section 2.2.1), except that the expectation of the predictive entropy term,  $\mathbb{E}_{\theta \sim p(\theta \mid \mathcal{D}_{\text{train}})}[\mathbb{H}[\mathbf{y}_* \mid \mathbf{x}_*, \theta]]$ , is computed exactly by marginalising over the variational depth posterior. In the case of regression models,  $\alpha_{\text{BALD}}$  can be computed exactly for DUNs, using the **epistemic uncertainty** component of the total variance:

$$\sigma_a^2 = \sum_{i=0}^D q(d = i) f(\mathbf{x}_*, d = i)^2 - \mu_a^2 + \sigma^2, \quad (2.38)$$

where  $\mu_a = \sum_{i=0}^D f(\mathbf{x}_*, d = i) q(d = i)$ .  $\alpha_{\text{Entropy}}$  is approximated as described in section 2.2.1 by artificially binning repeated samples from the predictive posterior into a histogram, but with DUNs the samples are selected from the set of per-layer predictions, with sampling weights equal to the posterior probabilities over depth.

## 2.3 Bias in active learning

To conclude this chapter, we address an acknowledged but relatively poorly understood problem in active learning: that actively selecting informative training points introduces bias to the inferences, as the training data no longer follow the population distribution (Dasgupta and Hsu, 2008; MacKay, 1992a). Given that many statistical results rely on data points being identically and independently distributed (i.i.d.) samples from the population distribution, applying standard estimators to actively sampled datasets results in optimising for the wrong objective (Farquhar et al., 2021). In an informal literature survey, Farquhar et al. (2021) find that, of 15 recent papers in active learning, only two acknowledge the presence of this sampling bias, and none implement measures to correct for it. Earlier proposed

solutions—e.g., those by Beygelzimer et al. (2009); Chu et al. (2011); Cortes et al. (2019); Ganti and Gray (2012)—are limited to specific modelling setups (for example, online rather than pool-based active learning) and are not generally applicable.

### 2.3.1 Unbiased risk estimators

We consider a method proposed in recent work by Farquhar et al. (2021) designed to correct for the bias induced by active sampling, which is agnostic to the active learning setup, model type and acquisition function used. Given a loss function  $\mathcal{L}(\mathbf{y}, f_\theta(\mathbf{x}))$ , an aim of inference is to find  $\theta$  that minimises the population risk  $r$  over all data points belonging to the population distribution  $p_{\text{data}}(\mathbf{y}, \mathbf{x})$ :

$$r = \mathbb{E}_{\mathbf{x}, \mathbf{y} \sim p_{\text{data}}} [\mathcal{L}(\mathbf{y}, f_\theta(\mathbf{x}))]. \quad (2.39)$$

Since the population risk cannot be computed exactly, it is approximated using a dataset of size  $N_{\text{train}}$  sampled from the population distribution. This yields the empirical risk,

$$\hat{R} = \frac{1}{N} \sum_{n=1}^{N_{\text{train}}} \mathcal{L}(\mathbf{y}_n, f_\theta(\mathbf{x}_n)), \quad (2.40)$$

an unbiased estimator of  $r$  when the data are drawn i.i.d. from  $p_{\text{data}}$ .

In the active learning setting, the risk is instead computed for a subset of  $M$  actively sampled data points:

$$\tilde{R} = \frac{1}{M} \sum_{m=1}^M \mathcal{L}(\mathbf{y}_m, f_\theta(\mathbf{x}_m)). \quad (2.41)$$

As explained above,  $\tilde{R}$  is not an unbiased estimator of either  $r$  or  $\hat{R}$ . An unbiased alternative,  $\tilde{R}_{\text{PURE}}$  (“plain unbiased risk estimator”), can be constructed by enforcing that each term individually is an unbiased estimator of  $r$ , leading to the formulation

$$\tilde{R}_{\text{PURE}} \equiv \frac{1}{M} \sum_{m=1}^M a_m; \quad a_m \equiv w_m \mathcal{L}_{i_m} + \frac{1}{N_{\text{train}}} \sum_{t=1}^{m-1} \mathcal{L}_{i_t}, \quad (2.42)$$

where the loss  $\mathcal{L}_{i_m} \equiv \mathcal{L}(y_{i_m}, f_\theta(x_{i_m}))$ , weights are defined as  $w_m \equiv \frac{1}{N_{\text{train}} \alpha(i_m; i_{1:m-1}, \mathcal{D}_{\text{pool}})}$  and  $i_m \sim \alpha(i_m; i_{1:m-1}, \mathcal{D}_{\text{pool}})$ . Intuitively, the estimator works by re-weighting each example’s contribution to the total loss by the inverse probability of that example having been selected, such that unusual examples with a particularly high probability of selection contribute proportionally less to the total risk. The proposal distribution  $\alpha(i_m; i_{1:m-1}, \mathcal{D}_{\text{pool}})$  is comparable to the acquisition functions discussed in section 2.1.1, except that it defines a *probability*

*mass* (rather than information scores) over the points in  $\mathcal{D}_{\text{pool}}$ , representing the probability of each index being sampled next, given  $\mathcal{D}_{\text{train}}$  contains indices  $i_{1:m-1}$ . The only limitation on  $\alpha(\cdot)$  is that it must assign non-zero probability to all indices in  $\mathcal{D}_{\text{train}}$ , a condition required to guarantee that  $\tilde{R}_{PURE}$  is unbiased and consistent. The acquisition functions  $\alpha_{\text{Entropy}}$  and  $\alpha_{\text{BALD}}$  can be easily converted to acquisition proposal distributions by passing the scores through a softmax function.

Farquhar et al. (2021) show that  $\tilde{R}_{PURE}$  is an unbiased estimator of  $r$ , a conclusion following from the observation that for each term  $a_m$ ,  $\mathbb{E}[a_m] = r$ .  $\tilde{R}_{PURE}$  is also proven to be consistent. The authors note that naïve important sampling,  $\frac{1}{M} \sum_{m=1}^M w_m \mathcal{L}_{i_m}$ , does not guarantee either of these properties.

In order to address certain pathologies of  $\tilde{R}_{PURE}$ —for example, that under uniform sampling, points sampled earlier are more highly weighted than later ones—a supplementary “levelled unbiased risk estimator”  $\tilde{R}_{LURE}$  is introduced:

$$\begin{aligned} \tilde{R}_{LURE} &\equiv \frac{1}{M} \sum_{m=1}^M v_m \mathcal{L}_{i_m} ; \\ v_m &\equiv 1 + \frac{N_{\text{train}} - M}{N_{\text{train}} - m} \left( \frac{1}{(N_{\text{train}} - m + 1) \alpha(i_m; i_{1:m-1}, \mathcal{D}_{\text{pool}})} - 1 \right). \end{aligned} \quad (2.43)$$

This estimator ensures that  $\mathbb{E}[v_m] = 1 \forall m, M, N, \alpha(i_m; i_{1:m-1}, \mathcal{D}_{\text{pool}})$ ; that is, that weights do not depend on the order in which data points are sampled. The resulting estimator also has lower variance than  $\tilde{R}_{PURE}$  (in practice, the experiment results in Farquhar et al. (2021) indicate that both estimators perform very similarly).

### 2.3.2 Interaction of biases in active learning

Interestingly, the effect of applying these unbiased risk estimators is found to differ depending on the class of model used. For overparameterised models such as neural networks, using  $\tilde{R}_{PURE}$  or  $\tilde{R}_{LURE}$  has no impact or is in fact detrimental to predictive performance. In contrast, underparameterised models such as linear regression benefit from the use of the unbiased estimators.

Figure 2.4 from Farquhar et al. (2021) illustrates the linear regression case with a toy example. In active learning, unusual data points are disproportionately sampled, resulting in the line trained with  $\tilde{R}$  being biased. Correcting for this sampling bias with  $\tilde{R}_{PURE}$  or  $\tilde{R}_{LURE}$  results

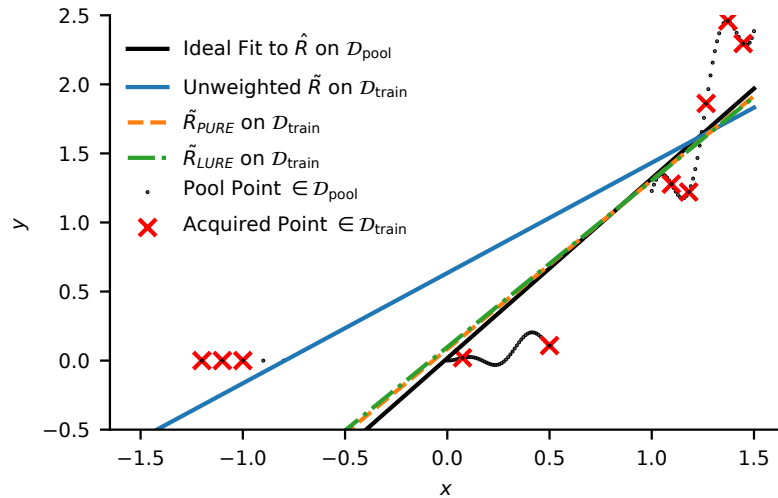


Figure 2.4: Illustrative linear regression. Active learning deliberately over-samples unusual points (red x’s), which no longer match the population (black dots). The biased estimator  $\tilde{R}$  puts too much emphasis on unusual points, resulting in a suboptimal model fit. The unbiased estimators  $\tilde{R}_{PURE}$  and  $\tilde{R}_{LURE}$  correct this bias, learning a function nearly equal to the ideal model that would be attained if training on a fully labelled  $\mathcal{D}_{pool}$  (Farquhar et al., 2021).

in a fit that is much closer to that obtained when training on all of  $\mathcal{D}_{pool}$ . Bias-correcting estimators are useful for inflexible models like linear regression because each data point affects the model globally, such that the input data distribution and exactness of the loss function are critical in distinguishing between potential model fits.

To explain the observation that correcting for sampling bias has a negative or negligible effect on performance for neural networks, note that differences between training and testing data typically result in some degree of overfitting in overparameterised models. Farquhar et al. (2021) observe that the overfitting bias tends to have an opposite sign to the active learning-induced bias—active learning encourages the selection of “difficult” to predict data points, increasing the overall risk, whereas overfitting results in the total risk being underestimated. The active learning and overfitting biases thus offset each other to a certain extent, and consequently, correcting for active learning bias does not improve model performance. Farquhar et al. (2021) claim that, in this sense, active learning can be considered as a form of regularisation for this class of model.

# 3

## Hypotheses: addressing biases in active learning with DUNs

*Sometimes a simple model will outperform a more complex model . . . Nevertheless, I believe that deliberately limiting the complexity of the model is not fruitful when the problem is evidently complex. Instead, if a simple model is found that outperforms some particular complex model, the appropriate response is to define a different complex model that captures whatever aspect of the problem led to the simple model performing well.*

– Radford Neal, *Bayesian Learning for Neural Networks*

Defining a model of appropriate complexity for a given problem is a challenging task, particularly in the context of active learning. If the underlying dataset is inherently complex, a well-performing model should typically reflect this complexity. Active learning, on the other hand, implies working with limited amounts of labelled data, on which complex models are prone to overfitting and poor generalisation. This chapter introduces the empirical approaches used in this thesis to explore the question of whether such a trade-off can be avoided or ameliorated through the use of DUNs in active learning. We motivate the use of these methods by a series of hypotheses about how they may help to address the issues of model complexity, overfitting bias, and other challenges in active learning.

### 3.1 DUNs for active learning

A primary objective of this thesis is to explore the application of DUNs to active learning and to compare their performance to that of BNNs trained with MFVI and MCDO. We posit that the ability to learn posterior probabilities over depth enables DUNs to adapt the model’s complexity to the changing dataset size during active learning, and thus to potentially outperform MFVI and MCDO, a hypothesis that is tested in section 4.3. It might be expected that simple, inflexible models will overfit to a lesser extent and achieve better generalisation performance at the start of active learning when the training set contains only a few data points. As the training set becomes larger and more interesting, the model may need to become deeper and more complex in order to explain the data. In a DUN this could easily be achieved by inferring networks of different depths according to the size of the dataset. For BNNs in weight-space, in contrast, the network depth is fixed, with the only recourse for influencing model complexity being through the learned weight posterior parameters. In this case, it is not clear how (or if) this mechanism would work.

#### Hypothesis 1

The ability to infer the depth of the network is a useful property for active learning that enables DUNs to minimise overfitting bias in the early stages of active learning, whilst still allowing the flexibility to learn more complex models as the volume of training data increases.

As noted by Antorán et al. (2020), the types of functions learnt by the DUN can also be influenced by manipulating the prior probabilities  $\beta_i$ . Selecting larger values of  $\beta_i$  for smaller depths encourages the DUN to behave like a simpler, less flexible model, which may be advantageous at the beginning of active learning to avoid overfitting. As the dataset increases in size, the likelihood component of  $p(d \mid \mathcal{D}_{\text{train}}; \theta)$  in equation (2.35) becomes more influential relative to the prior, allowing deeper and more complex networks to be learnt in accordance with complexity of the training set. This hypothesis is evaluated in section 4.4.4.

#### Hypothesis 2

The prior over depth can be leveraged as a regularisation mechanism to further reduce overfitting when training sets are small.

## 3.2 Addressing misspecification bias

Related to the idea of model complexity is that of model misspecification. Complex models such as deep neural networks tend to be underspecified, meaning that it is possible to find several predictors that perform equivalently well on held-out data. In other words, complex models are capable of making a greater variety of predictions (MacKay, 2003), implying that model uncertainty estimates will explode outside the convex hull of the data. This has been shown to be particularly true for ensemble models like DUNs (see, for example, figure 4.10) (Antorán et al., 2020). This behaviour is undesirable in active learning, as it means that examples are acquired from the extremities of the observed data range (see figure 4.3 for an example). Repeated acquisition at the extremities is often suboptimal, since what happens in these regions is typically less interesting than what happens in regions where data are more frequently observed (MacKay, 1992a). In addition, in batch-mode acquisition, this behaviour results in batches of correlated examples being acquired from clusters of exceptionally high-variance points.

We introduce a modification to the standard BALD acquisition strategy that aims to mitigate model misspecification bias. It applies to regression problems for which the output is standardised, such that the output has unit variance. As argued in section 2.1.1, maximising the mutual information is equivalent to maximising the model variance for regression problems. Given our knowledge that the data are standardised, predictive variances much larger than one can be attributed to model misspecification. We can thus correct for the bias by truncating the variances to a maximum value of one before using the values for acquisition:

$$\alpha_{\text{BALDtrunc}}(\mathbf{x}; \boldsymbol{\theta}, \mathcal{D}_{\text{train}}) = \begin{cases} 1 & \text{if } \alpha_{\text{BALD}} > 1 \\ \alpha_{\text{BALD}} & \text{otherwise.} \end{cases} \quad (3.1)$$

Where there are more candidates with model variance greater than one than the desired batch size, the batch members are sampled at random from this group. The effectiveness of this approach is evaluated in section 4.2.2.

### Hypothesis 3

Truncating BALD scores to the value of the maximum expected predictive variance helps to counteract the effects of model misspecification bias.

### 3.2.1 Correlation in batch acquisition

Although one expected outcome of addressing misspecification bias is that correlated acquisitions at the extremities are avoided, we also wish to address the issue of correlation in batch queries more generally. In some cases, correlation causes information-based acquisition to perform worse than random selection in batch-mode settings, as illustrated in section 4.2.1. In such scenarios, it is difficult to justify the use of active learning as a basis on which to measure the quality of model uncertainty estimates. We thus propose a simple stochastic relaxation of deterministic BALD acquisition, which aims to improve the diversity of the acquired batch (and ultimately the performance of the model) whilst being simpler to implement and more computationally efficient than the fully batch-aware methods presented in section 2.1.1.

Instead of taking the  $\text{argmax}$  of the scores given by  $\alpha_{\text{BALD}}(\cdot)$ , the scores are passed through the  $\text{softmax}$  function and candidate points are sampled with probability proportional to the  $\text{softmax}$  probabilities. The probabilities are tempered by multiplying the  $\alpha_{\text{BALD}}(\cdot)$  scores by a constant  $T$  before applying  $\text{softmax}$ . The magnitude of the temperature  $T$  controls how deterministic the resulting sampling is—a larger  $T$  corresponds to more certainly selecting the point with the highest BALD score, while a smaller  $\alpha_{\text{BALD}}(\cdot)$  implies closer to uniform sampling. This approach seeks to combine the informed acquisition of points, as measured by  $\alpha_{\text{BALD}}(\cdot)$ , with the diversity induced by random sampling, where the balance of diversity and informativeness is controlled by the temperature. This method is investigated in section 4.2.3.

#### Hypothesis 4

Introducing stochasticity to information-based selection improves the diversity of the acquired examples and helps to address loss of information due to correlation.

## 3.3 Investigating active learning bias

As part of the investigation into how DUNs operate in active learning settings, we examine the role of bias in further detail. Given our proposal that DUNs handle overfitting differently to other BNN methods, the experiments conducted by Farquhar et al. (2021) are relevant to this analysis, as they find that the effectiveness of eliminating bias induced by active selection is dependent on the magnitude of overfitting bias. We replicate the experiments from Farquhar et al. (2021) to quantify both active learning bias and overfitting bias (experimental implementation details are provided in section 4.4). We also apply the corrective weights



proposed in that paper and described in section 2.3.1 that aim to eliminate active learning bias from the loss function.

As discussed in section 2.3, Farquhar et al. (2021) find that correcting for statistical bias harms the performance of overparameterised models such as neural networks, as the statistical bias tends to offset the overfitting bias common to such models. We hypothesise that DUNs are a special case of neural network that may, in fact, benefit from the use of such corrective weights. In section 3.1 we argue that DUNs should naturally be able to handle smaller training set sizes at the beginning of active learning by inferring shallower networks, which are expected to overfit to a lesser extent than deeper, fixed-depth networks. In addition, we suggest that the prior probabilities over depth can be chosen so as to have a regularising effect on the complexity of the model. The work of Farquhar et al. (2021) on unbiased risk estimators is therefore potentially useful for active learning with DUNs, given that they find a positive effect of their corrective weights for inflexible models that overfit to a limited extent. This theory is explored in section 4.4.

**Hypothesis 5**

The overfitting bias that typically offsets sampling bias in neural networks is small or non-existent in DUNs with well-chosen prior probabilities over depth. As a result, removing the bias induced by active sampling positively impacts downstream performance for DUNs.



# 4

## Results

This chapter is concerned with the empirical evaluation of the methods described thus far, and in particular of the hypotheses presented in chapter 3. We begin by detailing the experimental setup and describing the relevant datasets in section 4.1. The following section investigates the impact of the active learning strategies introduced in section 2.1.1 on the performance of DUNs. In particular, we explore the problems of model misspecification bias and correlated batch queries and assess the effectiveness of the corrective modifications proposed in section 3.2. Section 4.3 compares the performance of DUNs to that of the baseline methods MCDO and MFVI. The final part of this chapter investigates bias in active learning, applying the methods described in section 2.3 to quantify types of bias and assess their impact on DUNs.

### 4.1 Experimental setup

#### 4.1.1 Data

We conduct experiments on three groups of datasets: toy regression datasets, the UCI regression datasets (Hernández-Lobato and Adams, 2015), and the MNIST image classification dataset (LeCun et al., 1998). Following Antorán et al. (2020), we use five synthetic, one-dimensional toy regression datasets as baselines. The first and second datasets, introduced in Antorán et al. (2020) and Izmailov et al. (2020), respectively, are each composed of three disjoint clusters of data with different arrangements. The third dataset is used in Foong et al. (2019) to assess the quality of uncertainty estimates produced by approximate inference methods in between clusters of data. The “Matern” dataset is generated by sampling a function from a Gaussian process with Matern kernel with additive Gaussian noise, while “Wiggle” is formed of samples from the function  $y = \sin(\pi x) + 0.2 \cos(4\pi x) - 0.3x + \varepsilon$  with

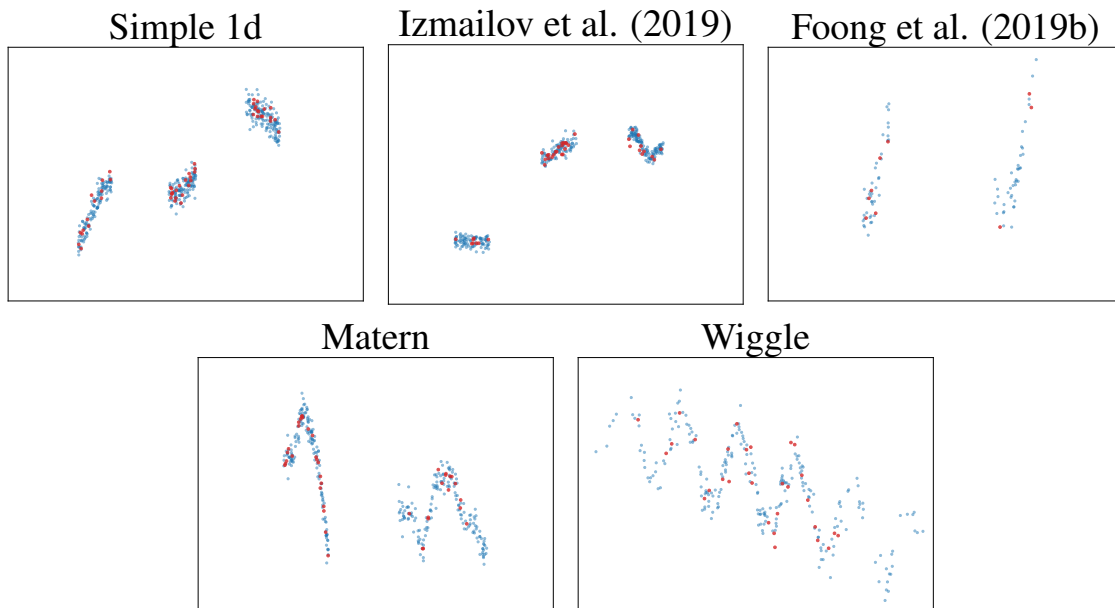


Figure 4.1: Data for toy datasets with example train-test split. Training data are indicated by blue points, test data by red points.

$\varepsilon \sim \mathcal{N}(0, 0.25)$  and  $x \sim \mathcal{N}(5, 2.5)$  (Antorán et al., 2020). The training data for these toy datasets are shown in figure 4.1. All methods are also evaluated on the UCI regression datasets, which have multi-dimensional inputs and are more reflective of realistic regression problems. A limited number of experiments are also carried out on the MNIST dataset in order to verify our conclusions on a larger scale classification problem.

As explained in section 2.1.1, we implement batch-based active learning, with batches of 10 (for MNIST and most toy datasets) or 20 (for most UCI datasets) data points acquired in each query, depending on the size of the dataset.<sup>1</sup> An initial training set representing a small proportion of the full data is selected uniformly at random in the first query. For all regression datasets, 80% of the data are used for training, 10% for validation and 10% for testing. The standard train-test split is used for MNIST. Details about dataset sizes, input dimensionality, and active learning specifications for each dataset are provided in table 4.1.

<sup>1</sup>All toy datasets have a query size of 10, except for Foong et al. (2019) with a query size of 5 as it is a smaller dataset. Likewise, all UCI regression datasets have a query size of 20, with the exception of Yacht with a query size of 10 as it is smaller. MNIST has a query size of 10, following Gal et al. (2017).

Table 4.1 Summary of datasets and active learning specifications. For the toy and UCI datasets, 80% of the data are used for training, 10% for validation and 10% for testing. For MNIST, the test and train set sizes are shown in parentheses, i.e., (train & test).

NAME	SIZE	INPUT DIM.	INIT. TRAIN SIZE	NO. QUERIES	QUERY SIZE
Simple 1d	501	1	10	30	10
Izmailov et al. (2020)	400	1	10	30	10
Foong et al. (2019)	100	1	10	15	5
Matern	400	1	10	30	10
Wiggle	300	1	10	20	10
Boston Housing	506	13	20	17	20
Concrete Strength	1,030	8	50	30	20
Energy Efficiency	768	8	50	30	20
Kin8nm	8,192	8	50	30	20
Naval Propulsion	11,934	16	50	30	20
Power Plant	9,568	4	50	30	20
Protein Structure	45,730	9	50	30	20
Wine Quality Red	1,599	11	50	30	20
Yacht Hydrodynamics	308	6	20	20	10
MNIST	70,000 (60,000 & 10,000)	784 (28 × 28)	10	10	10

### 4.1.2 Model specifications

For the regression problems, we implement a fully-connected network with residual connections, with 100 hidden nodes per layer. The networks contain 10 hidden layers for DUNs, or three hidden layers for MCDO and MFVI.<sup>2</sup> We use ReLU activations, and for DUNs batch normalisation is applied after every layer (Ioffe and Szegedy, 2015). Optimisation is performed over 1,000 iterations with full-batch gradient descent, with momentum of 0.9 and a learning rate of  $10^{-4}$ . A weight decay value of  $10^{-5}$  is also used. We do not implement early stopping, but the best model based on evaluation of the ELBO on the validation set is used for reporting final results. MFVI models are trained using five MC samples and the local reparameterisation trick explained in section 2.2.1, and prediction for both MCDO and MFVI is based on 10 MC samples. For MCDO models a fixed dropout probability of 0.1 is used. Unless otherwise specified, DUNs use a uniform categorical prior over depth, while MFVI networks use a  $\mathcal{N}(\mathbf{0}, I)$  prior over weights. These hyperparameter settings largely follow those used in Antorán et al. (2020) for the toy regression problems; a comprehensive hyperparameter search was not undertaken due to time and computational constraints, with the exception of the attempts outlined in section 4.3 to obtain a better fit for MFVI.

<sup>2</sup>Other depths were tested for the baseline methods, including single-layer networks, as well as the depth found to be optimal according to the DUN depth posterior, with similar results to the chosen depth of three layers. The results of these experiments are given in appendix A.4.

For image classification experiments we use the convolutional network architecture described in Antorán et al. (2020). The input block consists of a convolutional layer with  $5 \times 5$  kernel and a  $2 \times 2$  average pooling layer. The output block is comprised of a global average pooling layer, a linear layer followed by ReLU activation and batch normalisation, and a final linear layer. Predictions are made by passing the output of the linear layer through the softmax function. The intermediate blocks are bottleneck convolutional blocks described in He et al. (2016), with the outer number of channels chosen to be 64 and bottleneck channels set to 32. DUNs contain 10 intermediate blocks, while MCDO and MFVI networks contain three. Following PyTorch defaults for MNIST, optimisation is performed over 90 epochs, with the learning rate reduced by a factor of 10 at epochs 40 and 70. The initial learning rate is 0.1. We do not use a validation set to find the best model, instead reporting results on the final model. All other hyperparameters are as for the regression case.

All experiments are repeated 40 times with different weight initialisations and train-test splits for the toy and UCI regression datasets.<sup>3</sup> The standard MNIST train-test split is used for all image classification experiments. Unless otherwise specified, we report the mean and standard deviation of the relevant metric over the 40 experiment runs. All experiments are implemented in PyTorch (Paszke et al., 2019).<sup>4</sup>

## 4.2 Acquisition functions

This section examines the effectiveness of active learning with DUNs. We compare DUNs' performance on active learning problems with the maximum entropy  $\alpha_{\text{Entropy}}$  and BALD  $\alpha_{\text{BALD}}$  acquisition functions, described in section 2.1.1 and section 2.1.1 respectively, against a random acquisition baseline. Ideally, by taking into account model uncertainty, the active learning strategies should achieve optimal performance with significantly fewer labelled points than with random selection. Performance is evaluated in terms of test negative log-likelihood (NLL) and root mean-squared error (RMSE). For brevity, we report only NLL results for the regression problems in this section; similar conclusions can be drawn from the RMSE results, which can be found in appendix A.

---

<sup>3</sup>An exception is experiments on the Protein dataset, which are repeated only 30 times due to the cost of evaluation on the larger test set.

<sup>4</sup>Code is available at [https://github.com/chelsealuisa/DUN/tree/active\\_learning](https://github.com/chelsealuisa/DUN/tree/active_learning).

### 4.2.1 Toy regression with naïve batch acquisition

As an initial approach, we apply a simple extension of the active learning strategies to the batch acquisition case, by selecting the  $b$ -best examples ranked by  $\alpha_{\text{Entropy}}$  or  $\alpha_{\text{BALD}}$ . As shown in figure 4.2, this approach yields no improvement in learning efficiency over random acquisition, and in some cases performs worse than random selection. Potential causes for this finding include misspecification bias and correlation in batch acquisition, as discussed in section 3.2 and section 2.1.1, respectively. We now consider the techniques introduced in section 3.2 to mitigate these effects.<sup>5</sup>

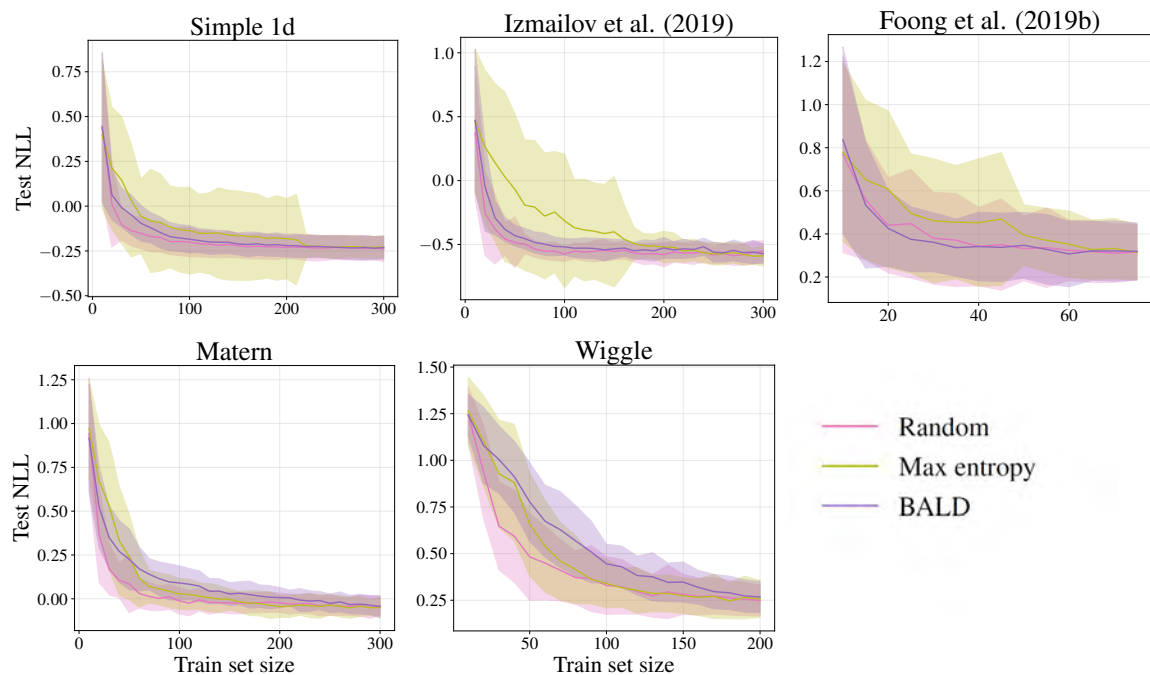


Figure 4.2: Test NLL vs. number of training points for DUNs evaluated on toy datasets. Maximum entropy and BALD acquisition functions are compared to a random acquisition baseline.

### 4.2.2 Truncated BALD

The first of these methods is truncated BALD, which caps the BALD scores at a maximum of one before computing the argmax of the scores. The objective is to counteract the effect of misspecification bias, which causes uncertainty estimates to explode in regions of the input

<sup>5</sup>Note that maximum entropy attains similar mean performance to BALD, but with much higher variance on the first three datasets. This is potentially due to the histogram approximation used to compute the predictive entropy for regression problems, which is dependent on hyperparameters such as the number of samples and number of bins used in the approximation, as discussed in section 2.1.1.

space beyond the most extreme points where data have been gathered (MacKay, 1992a). As illustrated in figure 4.3 for the Wiggle and Matern datasets, truncated BALD is effective in addressing this problem: in the right-hand column, the points selected under standard BALD acquisition (orange points) are exactly those with the most extreme  $x$ -values of the available unlabelled points. In the right-hand column, in contrast, using truncated BALD means that all candidate points with variance larger than one have an equal chance of being selected, resulting in the chosen points being better distributed across the high variance region (see in particular the positive end of the input range for Wiggle and the negative end for Matern).<sup>6</sup>

Unfortunately, truncated BALD has a negligible effect on overall performance, as shown in figure A.1 for the toy datasets and in figure 4.6 for the UCI regression datasets. To understand why truncation does not improve performance, we examine the initial steps of the querying process with truncated BALD acquisition in figure 4.4. As anticipated, the model variance exceeds unity only at the extremities of the input space, meaning that the batches acquired in the initial two steps (especially step two, figure 4.4b) are still chosen from the edges and thus remain somewhat correlated. By the third acquisition round, none of the BALD scores exceed one, such that this method is equivalent to standard BALD. As expected, standard BALD acquires clusters of correlated points (figure 4.4c and figure 4.4d), meaning that a lot of information about a localised region is added to the train set at the expense of acquiring information about other data-sparse regions. This explains why we observe that truncated BALD does not outperform standard BALD, and by extension random acquisition.<sup>7</sup>

---

<sup>6</sup>Note that on the left-hand side of the two Wiggle dataset plots, the model variance (blue shaded region) appears to be zero. Model variance is in fact very large in these regions, as shown in the acquisition function subplots; it does not appear in the upper plots because the mean function increases to infinity at too rapid a rate for the model variance to be visible. This is also true for figure 4.4 and figure 4.10.

<sup>7</sup>It should be noted that Gal et al. (2017) use batch acquisition on MNIST and find a significant performance improvement with BALD relative to random acquisition. They do not address the issue of correlated acquisitions, and it is not clear why their results appear to be unaffected by this phenomenon. Several other works also find a significant benefit for active learning (whether with BALD or other active acquisition strategies), however these implement serial acquisition (Hernández-Lobato and Adams, 2015; Houlsby et al., 2011; Kirsch et al., 2019).



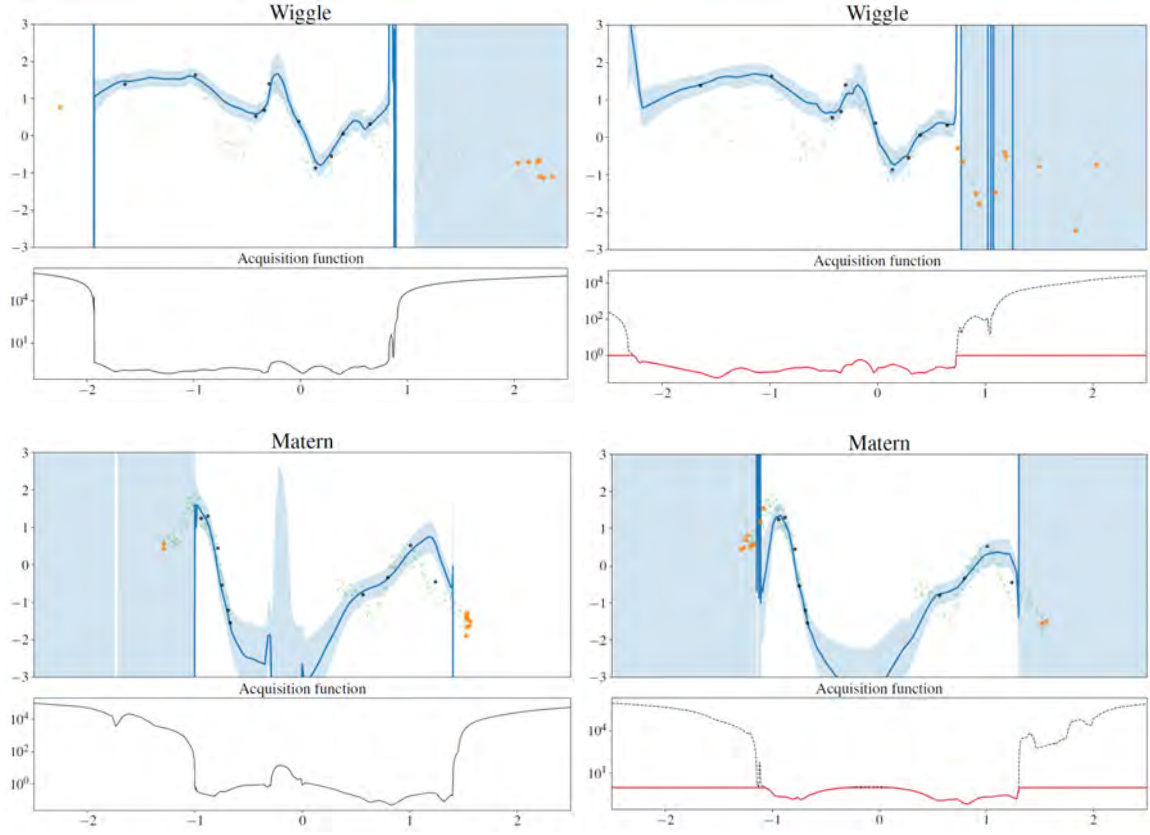


Figure 4.3: Illustration of the second acquisition step with standard BALD (left column) and truncated BALD (right column) acquisition functions, for Wiggle (top row) and Matern (bottom row) datasets. Black points denote the current labelled dataset, orange points the set of next acquired data points, and transparent green points the remainder of the pool set. Standard BALD acquires clusters of points at the extremities of the observed data range, while truncated BALD results in the acquired set being more spread-out across input space.

### 4.2.3 Stochastic BALD

Given the observation that truncated BALD is affected by correlation in the acquired batch, we next consider stochastic BALD,

$$\alpha_{\text{BALDstoch}}(\mathbf{x}; \theta, \mathcal{D}_{\text{train}}) = \text{softmax}(T \alpha_{\text{BALD}}(\mathbf{x}; \theta, \mathcal{D}_{\text{train}})) , \quad (4.1)$$

where candidates are sampled with probability equal to the resulting softmax probabilities. The temperature  $T$  controls how stochastic the selection is, as explained in section 3.2.1.

We evaluate the performance of  $\alpha_{\text{BALDstoch}}$  for several values of  $T$  on the UCI datasets in figure 4.5. We find that first truncating the BALD scores before applying softmax, i.e.,

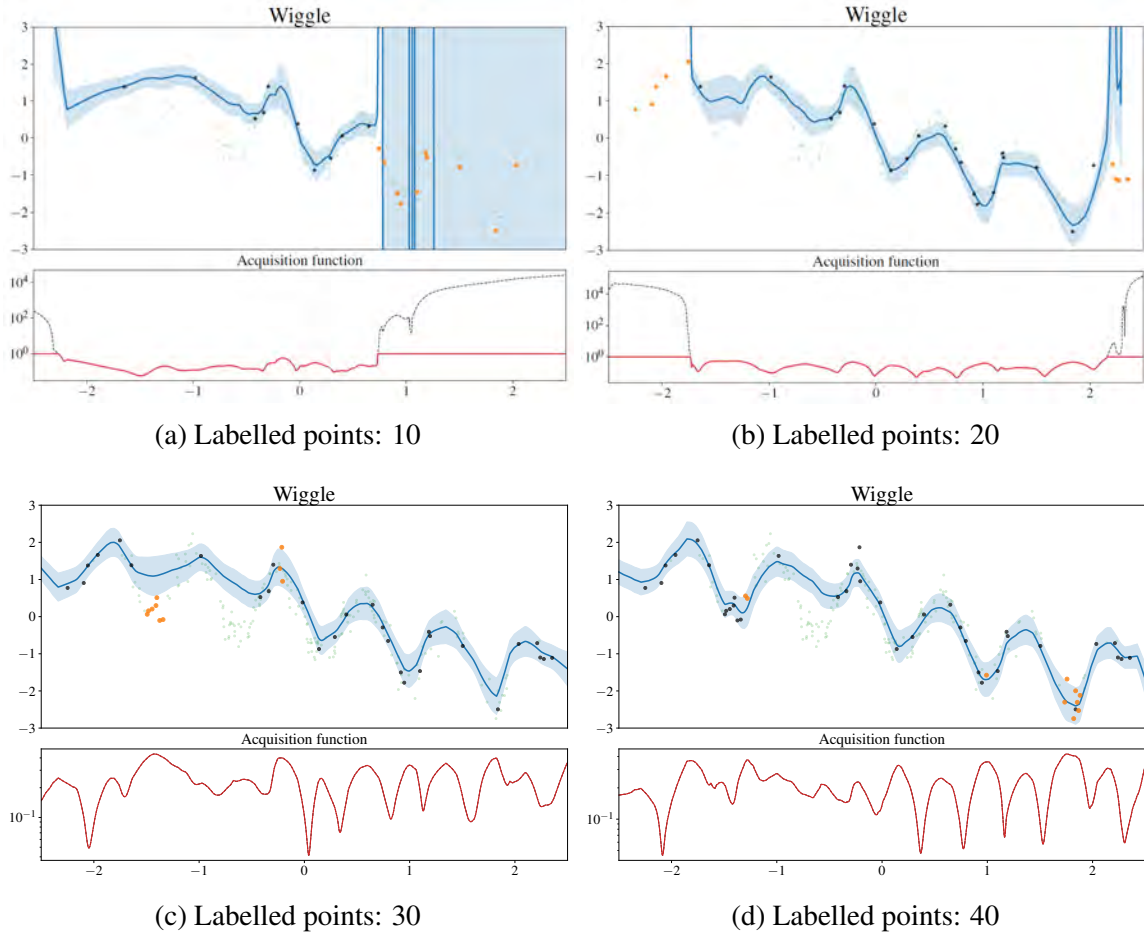


Figure 4.4: Illustration of the first four acquisition steps with truncated BALD acquisition for Wiggle dataset. Black points denote the current labelled dataset, orange points the set of next acquired data points, and transparent green points the remainder of the pool set. The acquisition function is plotted underneath the model fit (log scale), with the black dashed line representing BALD values and the red solid line truncated BALD.

$\alpha_{\text{BALDstoch}}(\mathbf{x}; \theta, \mathcal{D}_{\text{train}}) = \text{softmax}(T \alpha_{\text{BALDtrunc}}(\mathbf{x}; \theta, \mathcal{D}_{\text{train}}))$ , is important to avoid a “winner takes all” scenario—in which a single, very large BALD score dominates the softmax probabilities—in the initial acquisition steps, however has minimal impact on overall performance, as shown in figure A.2. For Boston, Kin8nm, Naval, Power, and Protein,  $T = 1$  and  $T = 10$  are found to be the best settings; for the other datasets, performance is similar for all or most values of  $T$ . Since  $T = 1$  underperforms slightly on Concrete, Energy, and Yacht, we proceed with  $T = 10$  as the optimal temperature.

Unlike truncated BALD, stochastic BALD with  $T = 10$  provides some improvement in performance for a number of the UCI datasets (namely, Kin8nm, Naval, Power and Protein),

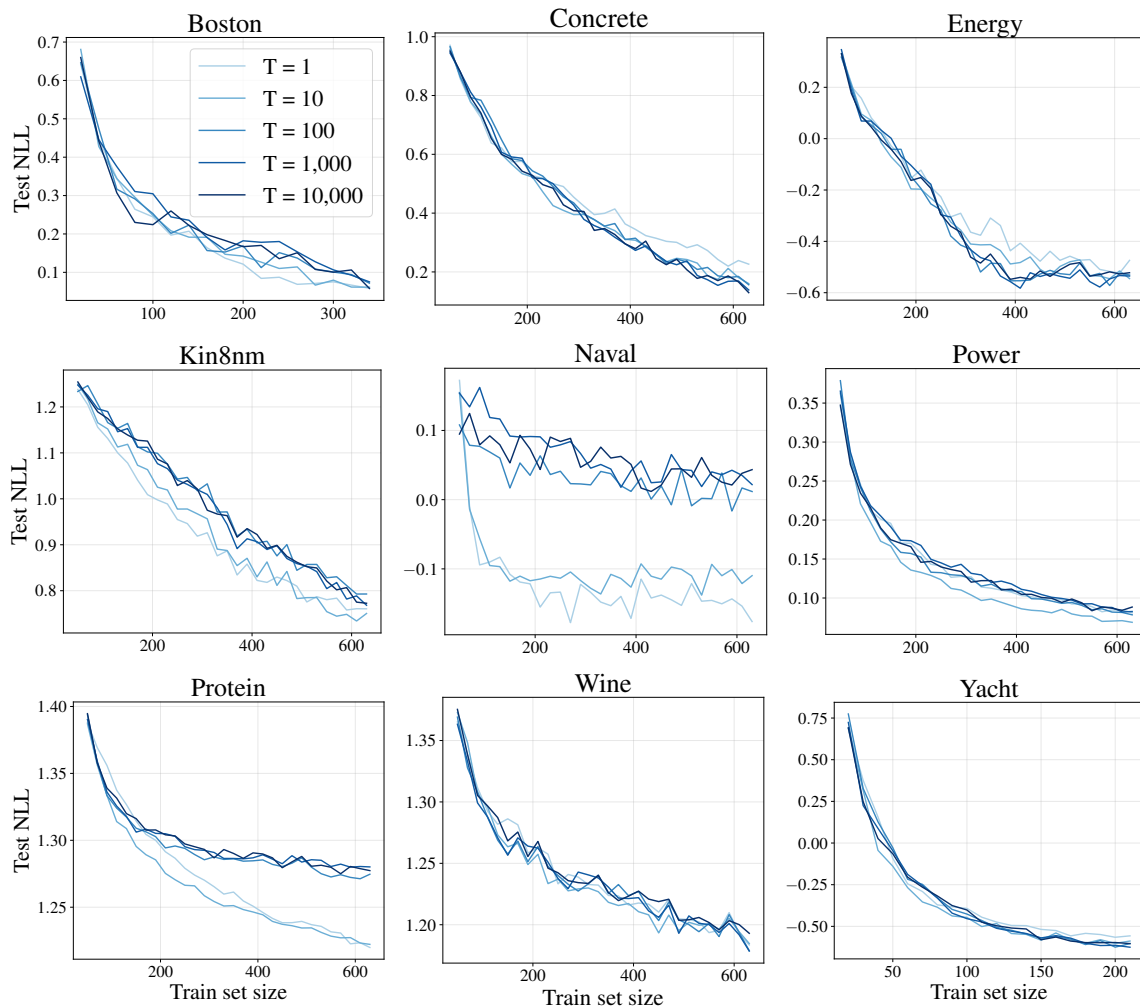


Figure 4.5: Test NLL of DUNs using a stochastic relaxation of BALD evaluated on UCI datasets. Different temperatures of the proposal distribution are compared. Means of 40 runs of the experiments are shown; standard deviations are not plotted for clarity.

as illustrated in figure 4.6 . In these cases, stochastic BALD effectively combines diversity of the acquired batch through sampling with informativeness of the individual points by leveraging the BALD scores. Figure 4.7 compares stochastic BALD to random and maximum entropy acquisition and confirms that stochastic BALD performs at least as well as, and for some datasets marginally better than, random selection (with the exception of Naval, for which random outperforms stochastic BALD).

### Image classification with stochastic BALD

To conclude the analysis of acquisition strategies, we compare their performance on a classification problem. Figure 4.8 displays the test accuracy and NLL when applying stochastic

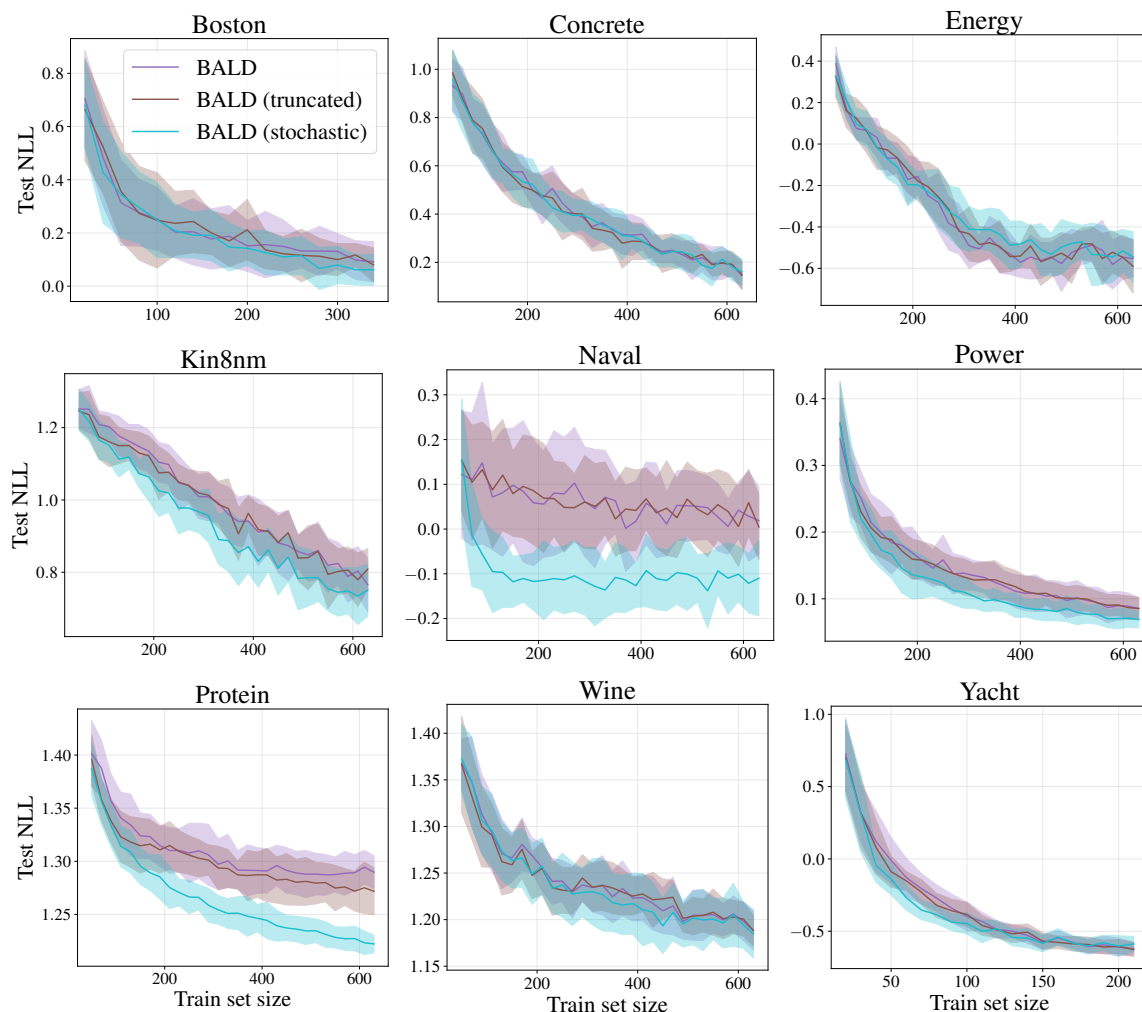


Figure 4.6: Test NLL vs. number of training points for DUNs evaluated on UCI datasets. Truncated and stochastic BALD acquisition functions are compared to standard BALD.

BALD, standard BALD, maximum entropy and random selection to the MNIST image classification dataset. As for the regression problems, stochastic BALD is approximately as effective as random selection. The benefit of the stochastic variant of BALD is clear when comparing to standard BALD, which is the poorest performing of the four methods. Interestingly, maximum entropy underperforms the BALD and random selection to a greater extent on the classification problem than in the regression setting, despite the fact that the entropy does not need to be approximated in the classification case.

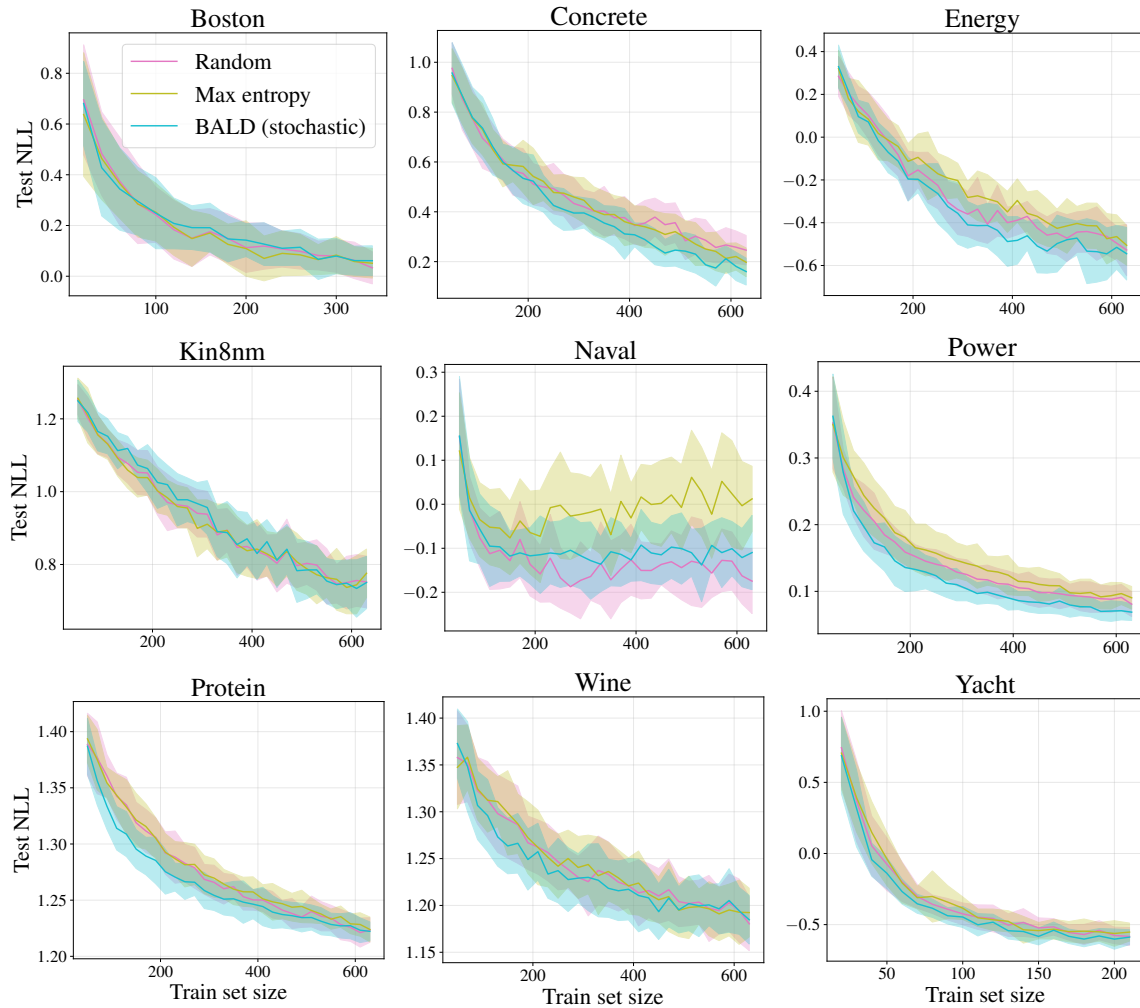


Figure 4.7: Test NLL vs. number of training points for DUNs evaluated on UCI datasets. Maximum entropy and stochastic BALD acquisition functions are compared to a random acquisition baseline.

## 4.3 DUN and baseline performance

This section compares the performance of DUNs to the baseline methods MCDO and MFVI. We again report only NLL results, but equivalent RMSE results are provided in appendix A.2. All results presented in this section are based on the stochastic BALD acquisition function, with truncation of  $\alpha_{BALD}$  prior to applying  $\text{softmax}$ , using temperature  $T = 10$ .

### 4.3.1 Toy datasets

Test NLL evaluated on the toy datasets is shown in figure 4.9. Performance for DUNs improves as the training set grows in size, as expected. This is not, however, always true for

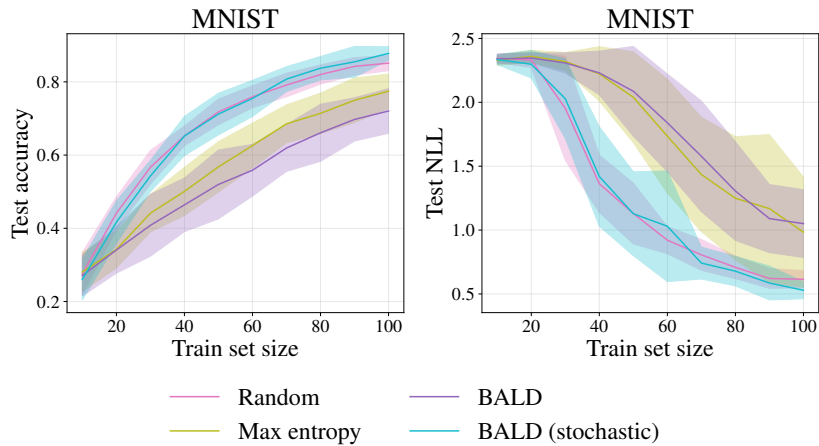


Figure 4.8: Test classification accuracy (left) and NLL (right) vs. number of training points for DUNs evaluated on MNIST. Maximum entropy, standard BALD, and stochastic BALD acquisition functions are compared to a random acquisition baseline.

MCDO and MFVI. MFVI performs poorly on all datasets independent of the training set size, suggesting that the models have underfit (this is also true for MCDO on the Matern and Wiggle datasets). Examining the fit of each of the methods on Matern and Wiggle in figure 4.10 (using the largest training set size from the final acquisition step) shows that this is the case—MFVI and MCDO severely underfit on both datasets, failing to capture nearly any variation in the data beyond a linear trend. DUNs, on the other hand, are able to fit the data well in the regions of observed data, and learn sensible uncertainty estimates outside these regions.

In an effort to improve the fit of the MFVI models, we repeat the experiments using twice as many optimisation steps for all three models, with the results shown in figure 4.11. The performance of MFVI does not change noticeably and remains substantially worse than that of both DUNs and MCDO. This may be attributable to variational over-pruning, as discussed in Trippe and Turner (2018).<sup>8</sup> The performance of MCDO improves on Matern, although is still significantly worse than the DUN, and remains poor on Wiggle. This is consistent with the results of Antorán et al. (2020), who find that MCDO underfits faster-varying functions, such as those required to fit the Matern and Wiggle datasets.

<sup>8</sup>We additionally repeat the experiments for MFVI with much smaller variance initialisations for the variational approximation, in an attempt to encourage a better fit to the data. This approach initialises training with variational distributions that approach delta functions around the mean, effectively yielding MAP estimation. This encourages a close fit to the data in the initial stages of training, which is later regularised by the KL term as training progresses. This should counteract the tendency of VI to optimise only for the KL term and explain variation in the data through noise. Unfortunately this approach yields poorer results than larger variance initialisations, so we do not present the results here.

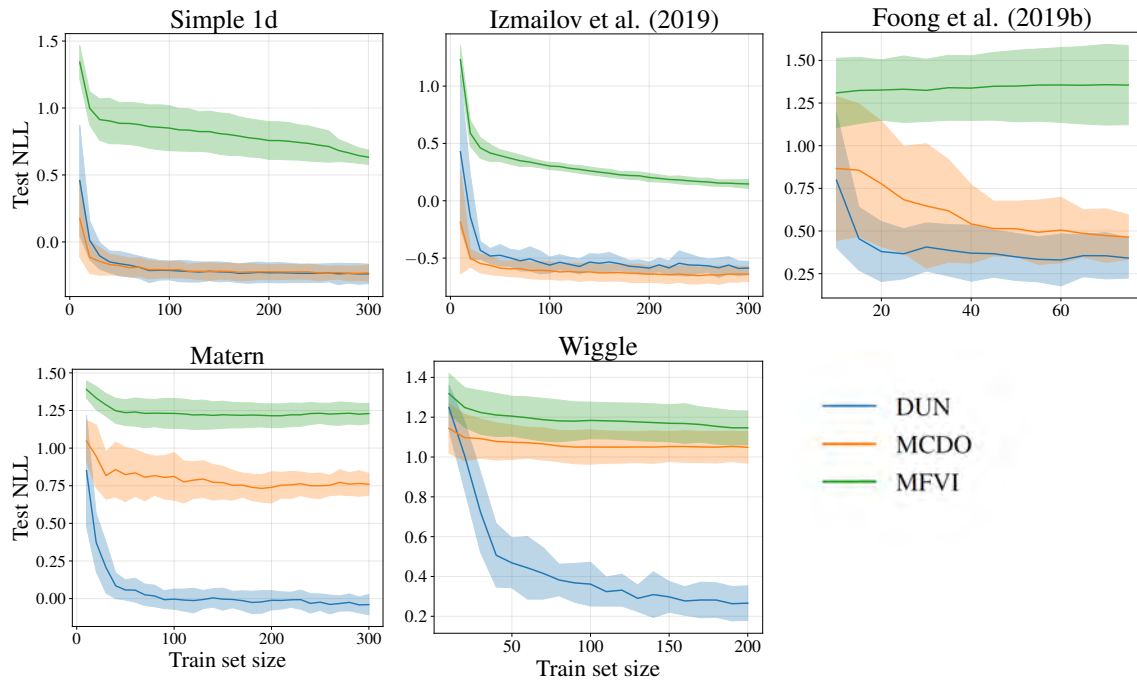


Figure 4.9: Test NLL vs. number of training points using stochastic BALD acquisition evaluated on toy datasets. The performance of DUNs, MCDO and MFVI is compared.

Hypothesis 1, proposed in section 3.1, is that DUNs are expected to adapt their inferred depth to the complexity of the dataset, learning shallower networks at the beginning of active learning and increasingly deep networks as more labelled data are acquired. Figure 4.12 compares the posterior probabilities over depth for DUNs trained on datasets from the first and final steps of active learning, and illustrates that this does occur in practice. Whether this feature of DUNs translates to meaningful performance advantages is difficult to conclude based on the toy dataset results, given that MFVI and MCDO suffer from underfitting. This question is explored further in section 4.3.2 in the context of the UCI datasets.

### 4.3.2 Tabular regression

We next compare the performance of DUNs, MCDO and MFVI on the UCI datasets. On these higher-dimensional datasets, the superiority of DUNs is not as clear as in the case of the toy datasets. In terms of both NLL (figure 4.13) and RMSE (figure A.7), DUNs clearly outperform the baselines on Concrete, Energy, Kin8nm, Naval and Protein, but achieve comparable performance to MCDO on Boston, Power, Wine and Yacht. MFVI exhibits the poorest performance on all of the datasets.

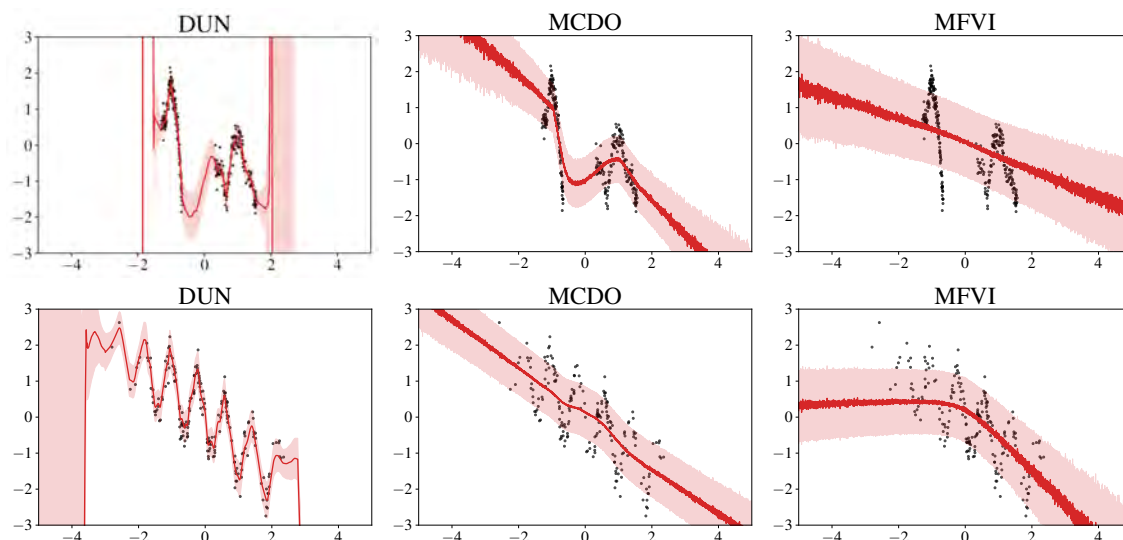


Figure 4.10: Model fit of a DUN, MCDO and MFVI on the Matern (top row) and Wiggle (bottom row) datasets.

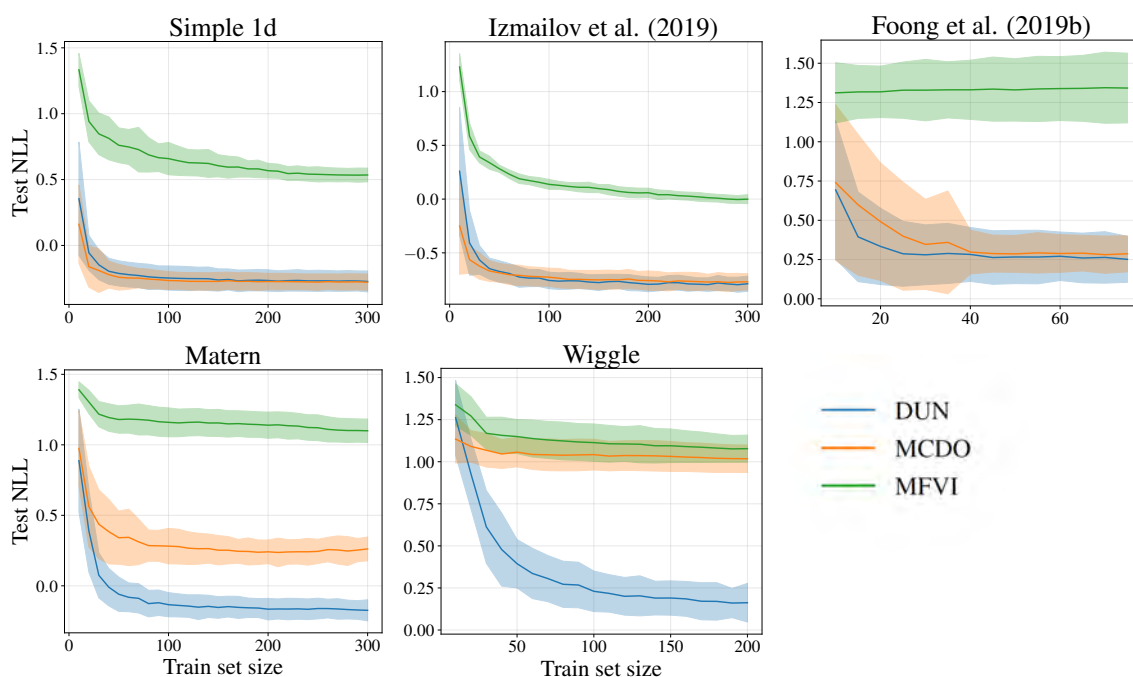


Figure 4.11: Test NLL vs. number of training points evaluated on toy datasets. DUNs, MCDO and MFVI models are trained using 2,000 epochs.

The observation that DUNs adapt the posterior probabilities over depth to the size of the training set is more evident with the UCI datasets than the toy datasets, as shown in figure 4.14. For Kin8nm, for example, layers two and three are assigned the highest posterior probabilities



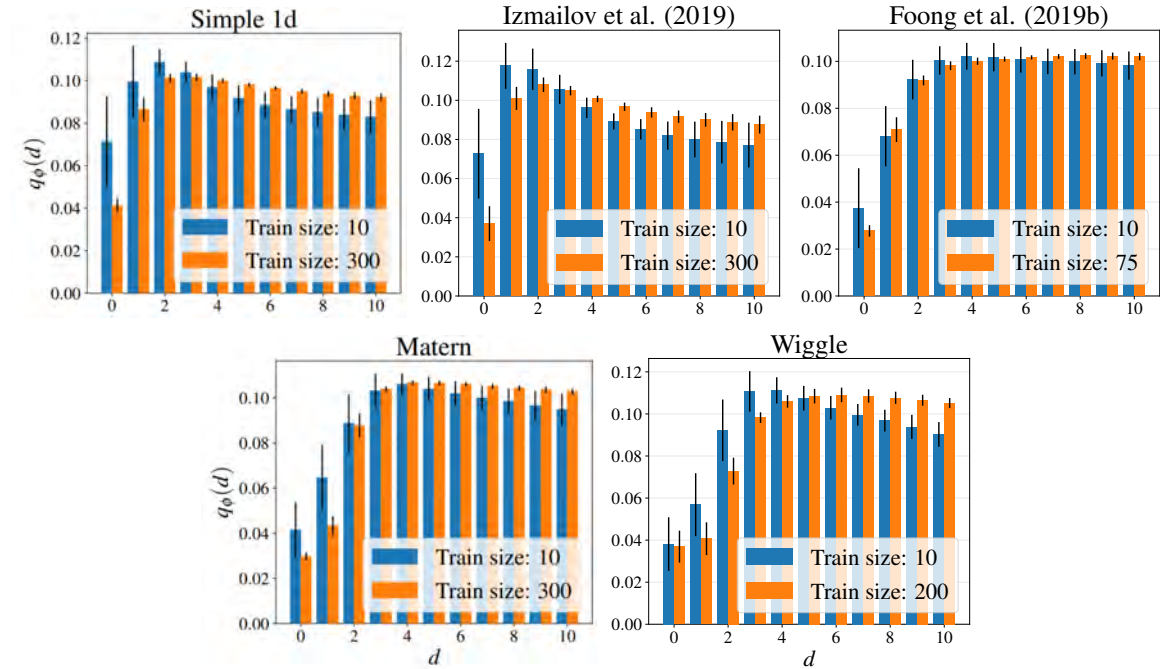


Figure 4.12: Posterior probabilities over depth for DUNs trained on toy datasets, for the smallest (blue bars) and largest (orange bars) labelled datasets used in active learning.

for a training set size of 50, with the probabilities declining as the depth increases. For a training set containing 630 points, in contrast, the posterior probabilities are largest for layers eight and above. Given the results from figure 4.13, it is plausible to conclude that DUNs’ flexibility over depth is useful in active learning settings (although this property does not result in DUNs consistently outperforming MCDO in all cases). DUNs’ performance may also be attributable to better quality uncertainty estimates, or to the fact that  $\alpha_{\text{BALD}}$  can be computed exactly for DUNs.

### 4.3.3 Image classification

Finally, we verify the conclusions drawn in this section on the MNIST classification problem. We focus on comparing DUNs to MCDO, as MFVI was shown to be less competitive on all of the regression problems. Figure 4.15a and figure 4.15b plot the accuracy and NLL for a DUN and MCDO network, with the DUN attaining marginally better performance than MCDO on both metrics (although not exceeding one standard deviation’s difference), which is consistent with the regression results. Also consistent with the regression setting is the fact that the posterior assigns more weight to deeper subnetworks as the training set grows in size, as shown in figure 4.15c.

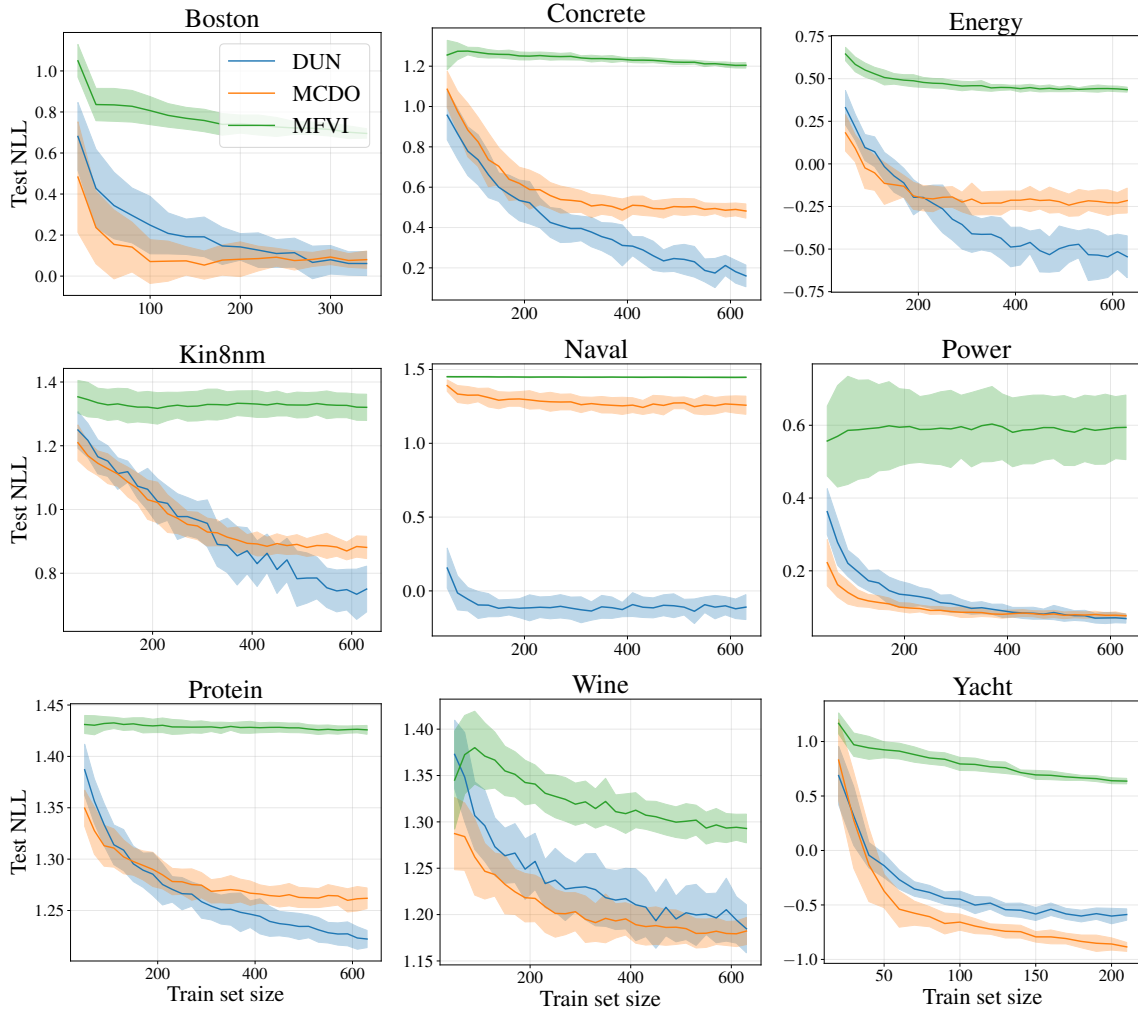


Figure 4.13: Test NLL vs. number of training points using stochastic BALD acquisition function evaluated on UCI datasets. DUNs, MCDO and MFVI are compared.

## 4.4 Bias in active learning

To conclude the chapter on empirical results, we explore the impact of bias and the interaction of different types of biases in active learning with DUNs. Section 2.3 introduces the concept of *active learning bias*, induced when active selection of points results in a training dataset that is not drawn from the population distribution. This is opposed to *overfitting bias*, which arises when evaluation of the model loss on the training data gives rise to dependency between the data and parameters  $\theta$ . The following analyses replicate the methods presented in Farquhar et al. (2021) to quantify active learning bias and overfitting bias in DUNs and MCDO models, and examine how their proposed corrective weights for active learning bias impact downstream performance. The section concludes by evaluating hypothesis 2

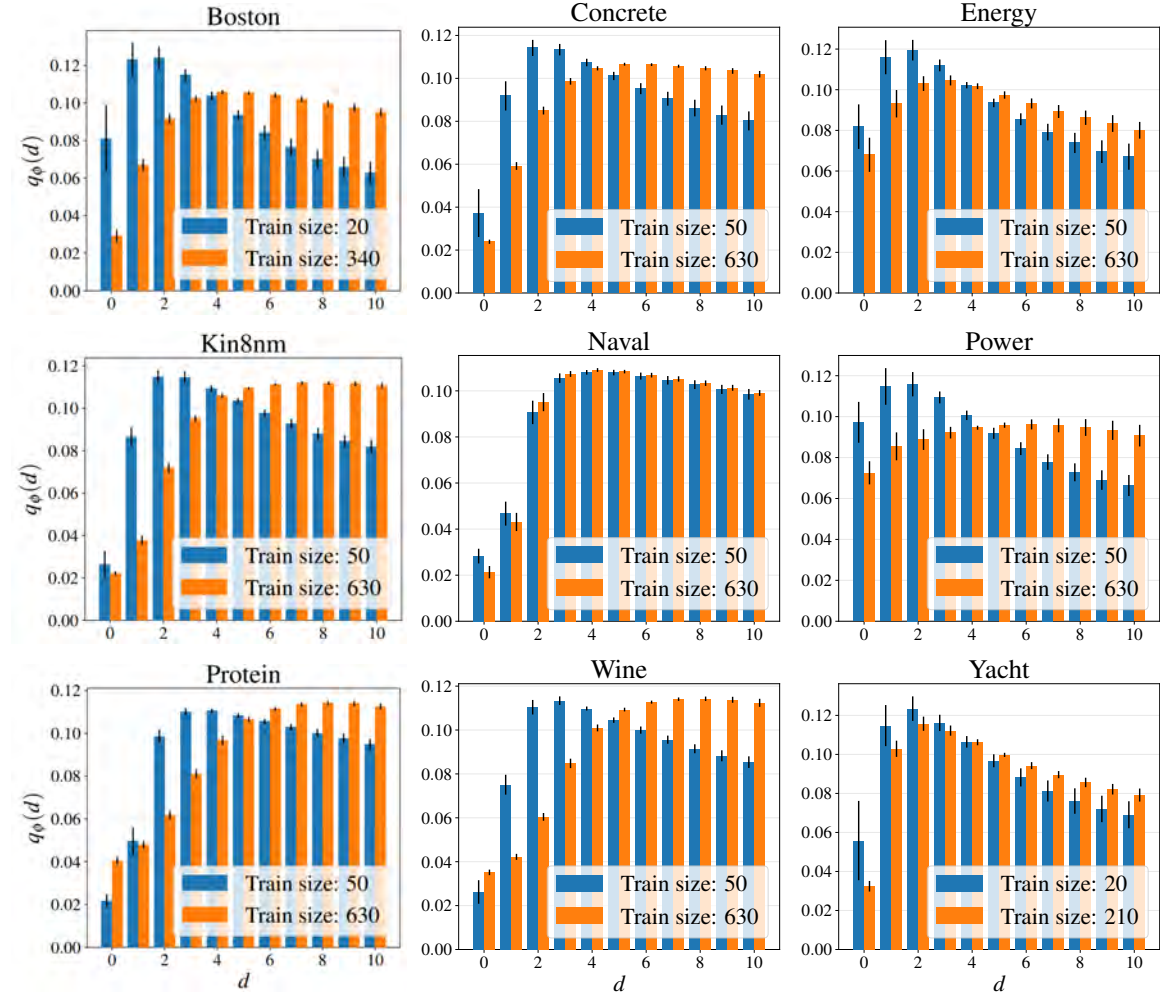


Figure 4.14: Posterior probabilities over depth for DUNs trained on UCI regression datasets, for the smallest (blue bars) and largest (orange bars) labelled datasets used in active learning.

(section 3.1), that the depth prior in DUNs can be used to influence model complexity and the subsequent degree of overfitting bias.

#### 4.4.1 Quantifying active learning bias

As a first exercise, we estimate the extent of active learning bias when using the standard risk estimator  $\tilde{R}$  (equation (2.40)), and verify that the unbiased risk estimator  $\tilde{R}_{LURE}$ , defined in equation (2.43), eliminates this bias. The experimental setup is as follows: a model is trained on 1,000 randomly selected data points from  $\mathcal{D}_{\text{pool}}$  using the standard NLL loss. Given fixed model parameters, evaluation points are actively sampled from a test set  $\mathcal{D}_{\text{test}}$ , using stochastic BALD acquisition with  $T = 10$ . After each evaluation point is acquired, the

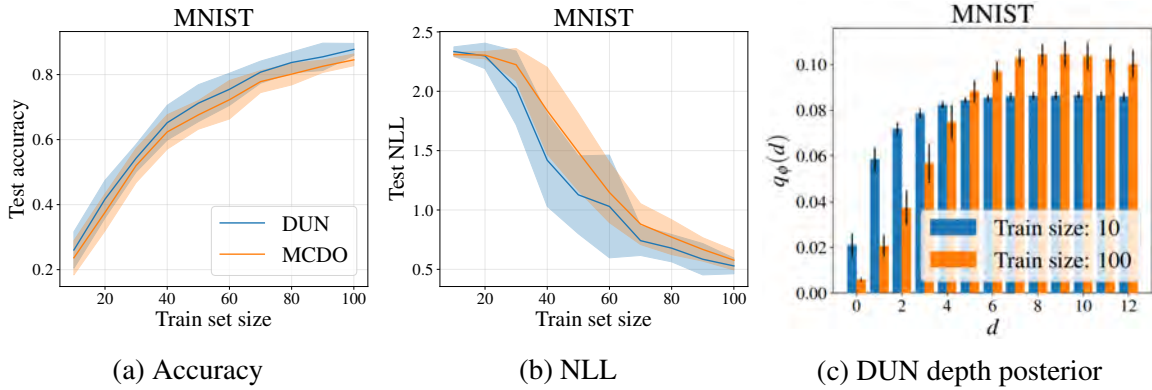


Figure 4.15: Test accuracy (left) and NLL (middle) vs. number of training points using stochastic BALD acquisition function evaluated on MNIST. The performance of DUNs and MCDO is compared. Right: DUN posterior probabilities over depth, for the smallest (blue bars) and largest (orange bars) labelled datasets used in active learning.

biased risk  $\tilde{R}$  and unbiased risk  $\tilde{R}_{LURE}$  are evaluated on the  $M$  actively sampled evaluation points. The active learning bias  $B_{ALB}$  is then estimated by subtracting  $\tilde{R}$  or  $\tilde{R}_{LURE}$  from the population risk  $r$ , which is approximated by evaluating the risk on the full  $\mathcal{D}_{\text{test}}$ :

$$B_{ALB}(\tilde{R}_{(\cdot)}) = r - E[\tilde{R}_{(\cdot)}]. \quad (4.2)$$

By computing both components of  $B_{ALB}$  using fixed model parameters  $\theta$  that are independent of  $\mathcal{D}_{\text{train}}$ , we abstract from any overfitting bias, enabling the bias induced by actively sampling the evaluation points to be isolated.

Figure 4.16 shows  $B_{ALB}(\tilde{R})$  and  $B_{ALB}(\tilde{R}_{LURE})$  for DUNs applied to the UCI datasets.  $B_{ALB}(\tilde{R}_{LURE})$  remains constant at nearly zero for all  $M$ , showing that the unbiased estimator  $\tilde{R}_{LURE}$  is effective in removing this source of bias. For the unweighted estimator  $\tilde{R}$ , in contrast, active learning bias is present whenever  $M < N$ . The bias is negative, since active learning tends to over-sample unusual or difficult to predict points, resulting in a higher estimated risk than the true risk.

We next compare active learning bias in DUNs to MCDO in figure 4.17. We present only the means over the 40 experiment runs without standard deviations for clarity. For MCDO  $\tilde{R}_{LURE}$  is also effective in removing active learning bias. The magnitude of the active learning bias with  $\tilde{R}$  is generally substantially smaller for MCDO than for DUNs (with the exception of the Yacht data). This implies that the points selected for labelling by DUNs tend to overrepresent unusual parts of the population distribution to a greater extent than those selected by MCDO.

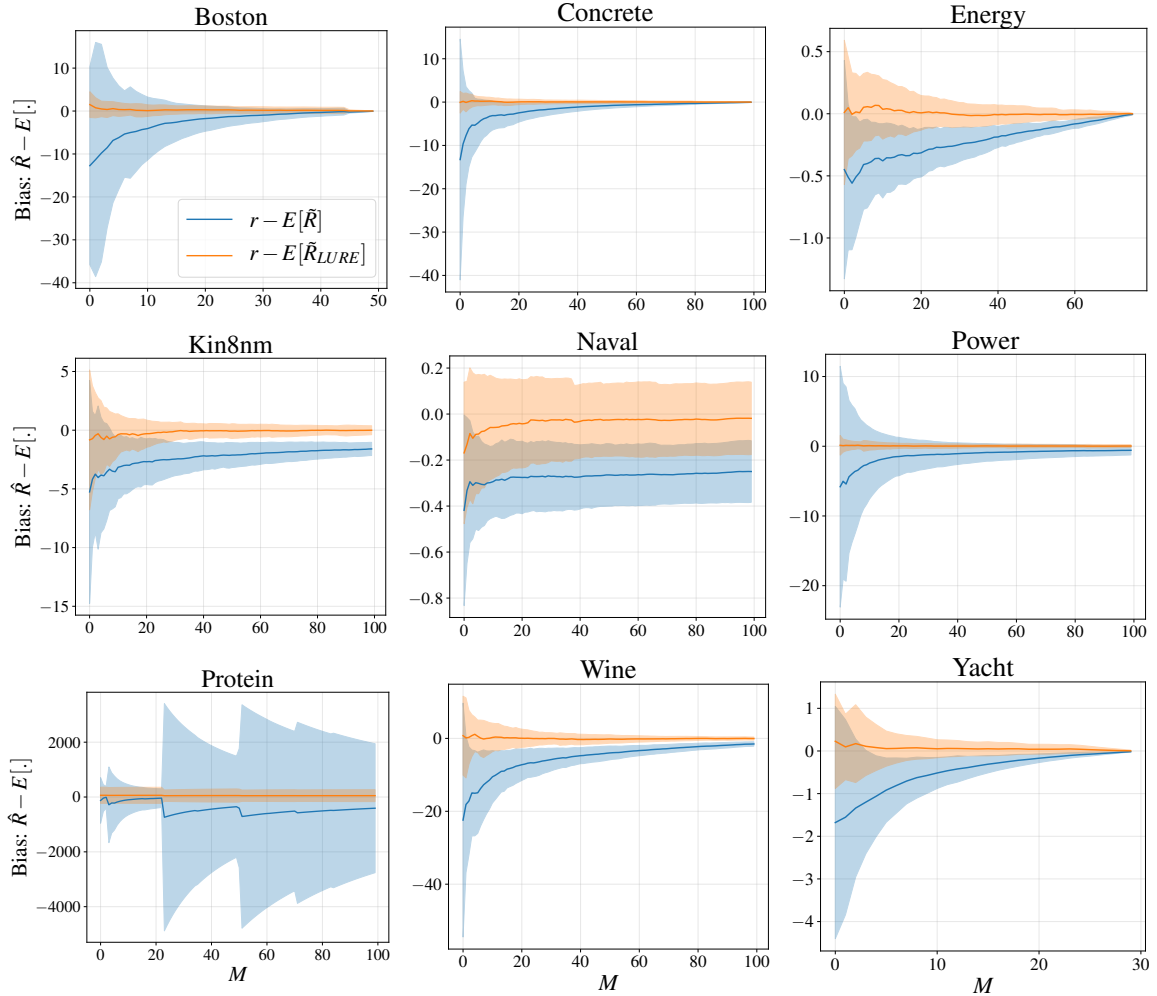


Figure 4.16: Bias introduced by actively sampling points with DUNs, evaluated using  $\tilde{R}$  (in blue) and  $\tilde{R}_{LURE}$  (in orange). The unbiased risk estimator  $\tilde{R}_{LURE}$  eliminates active learning bias, while the standard estimator  $\tilde{R}$  tends to overestimate risk.

#### 4.4.2 Training with unbiased risk estimators

Having confirmed that  $\tilde{R}_{LURE}$  eliminates active learning bias, we next seek to determine whether using the unbiased estimator during training improves downstream performance. This is achieved by using  $\tilde{R}$  and  $\tilde{R}_{LURE}$  as the training objectives in an active learning problem, and evaluating the NLL and RMSE for the resulting model on  $\mathcal{D}_{\text{train}}$ . In an effort to mirror the experimental approach of Farquhar et al. (2021) as closely as possible, we use serial rather than batch acquisition for the experiments presented in this section.

Intuitively, removing active learning bias by training with the weighted estimator should improve performance on the downstream task. Figure 4.18 shows, however, that using

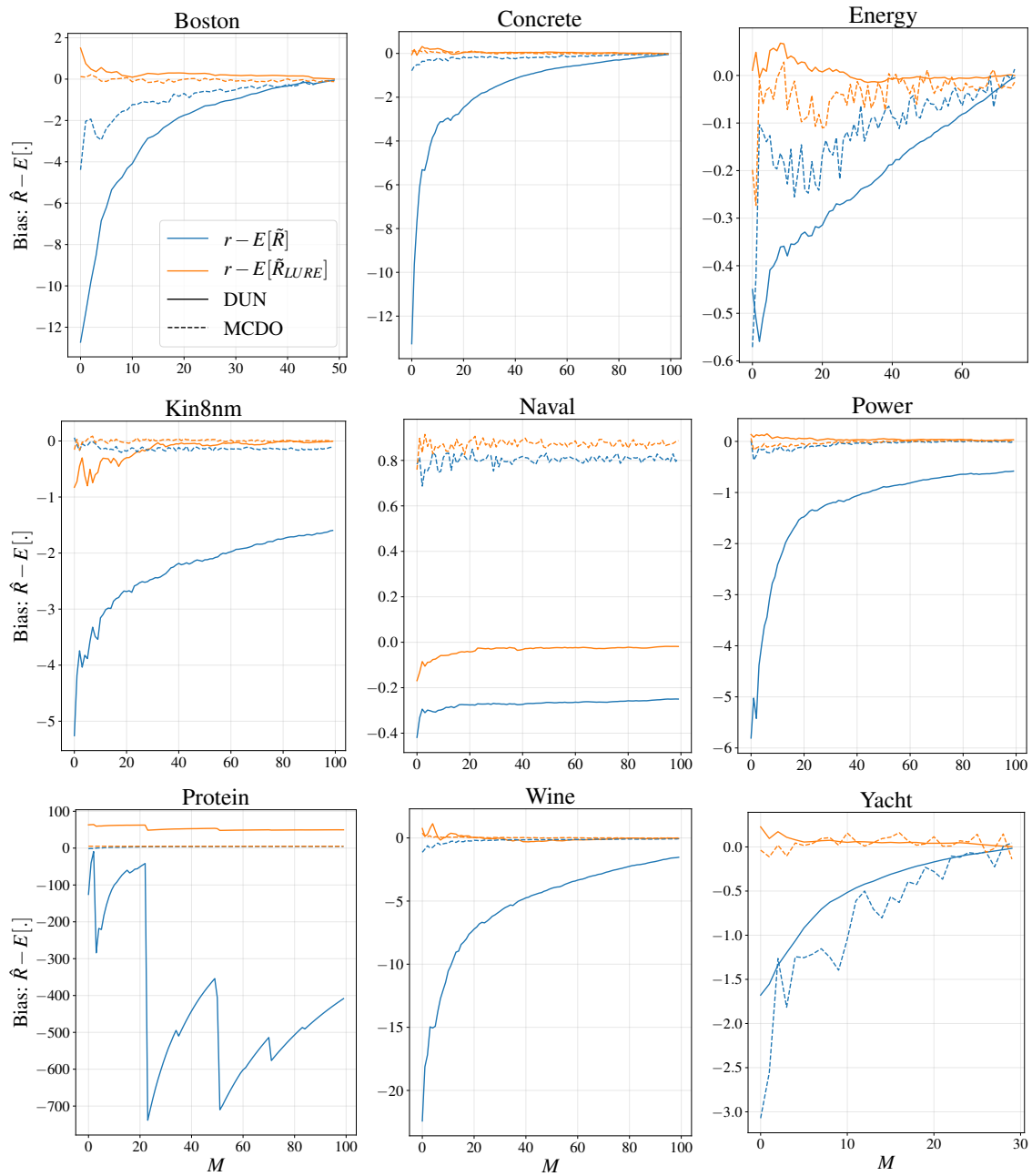


Figure 4.17: Active learning bias for DUNs (solid lines) and MCDO (dashed lines), evaluated using  $\tilde{R}$  (in blue) and  $\tilde{R}_{LURE}$  (in orange).

$\tilde{R}_{LURE}$  either does not affect or is to the detriment of NLL performance (equivalent RMSE results are provided in appendix A.3). Before moving to potential explanations for this result, we also consider the impact of  $\tilde{R}_{LURE}$  on MCDO in figure 4.19. Instead of plotting the NLL for models trained with  $\tilde{R}$  and  $\tilde{R}_{LURE}$  separately as in figure 4.18, we plot the difference  $NLL_{\tilde{R}} - NLL_{\tilde{R}_{LURE}}$ , where  $NLL_{\tilde{R}}$  is the NLL evaluated on a model trained with  $\tilde{R}$  and  $NLL_{\tilde{R}_{LURE}}$  is the equivalent for  $\tilde{R}_{LURE}$ . The difference is positive when the weighted estimator improves downstream performance and negative when it harms performance. As with DUNs, removing active learning bias has a negligible impact on downstream performance of MCDO for most datasets. Exceptions are Yacht, for which debiasing the estimator improves performance, and Protein, which has poorer performance with the unbiased estimator.

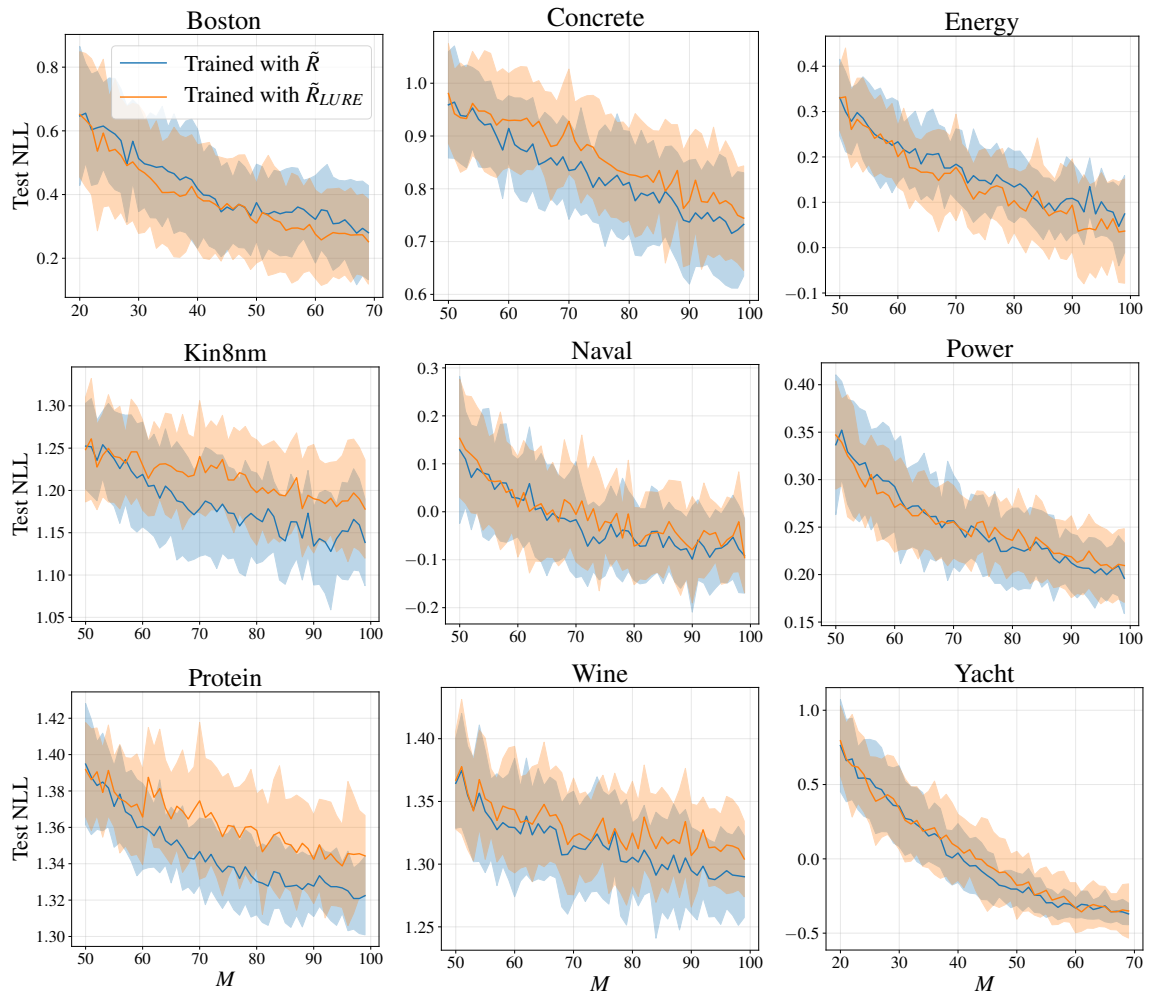


Figure 4.18: Test NLL for DUNs trained with  $\tilde{R}$  (in blue) and  $\tilde{R}_{LURE}$  (in orange) evaluated on UCI datasets. A larger value of the orange line indicates that training with  $\tilde{R}_{LURE}$  harms performance, despite removing active learning bias.

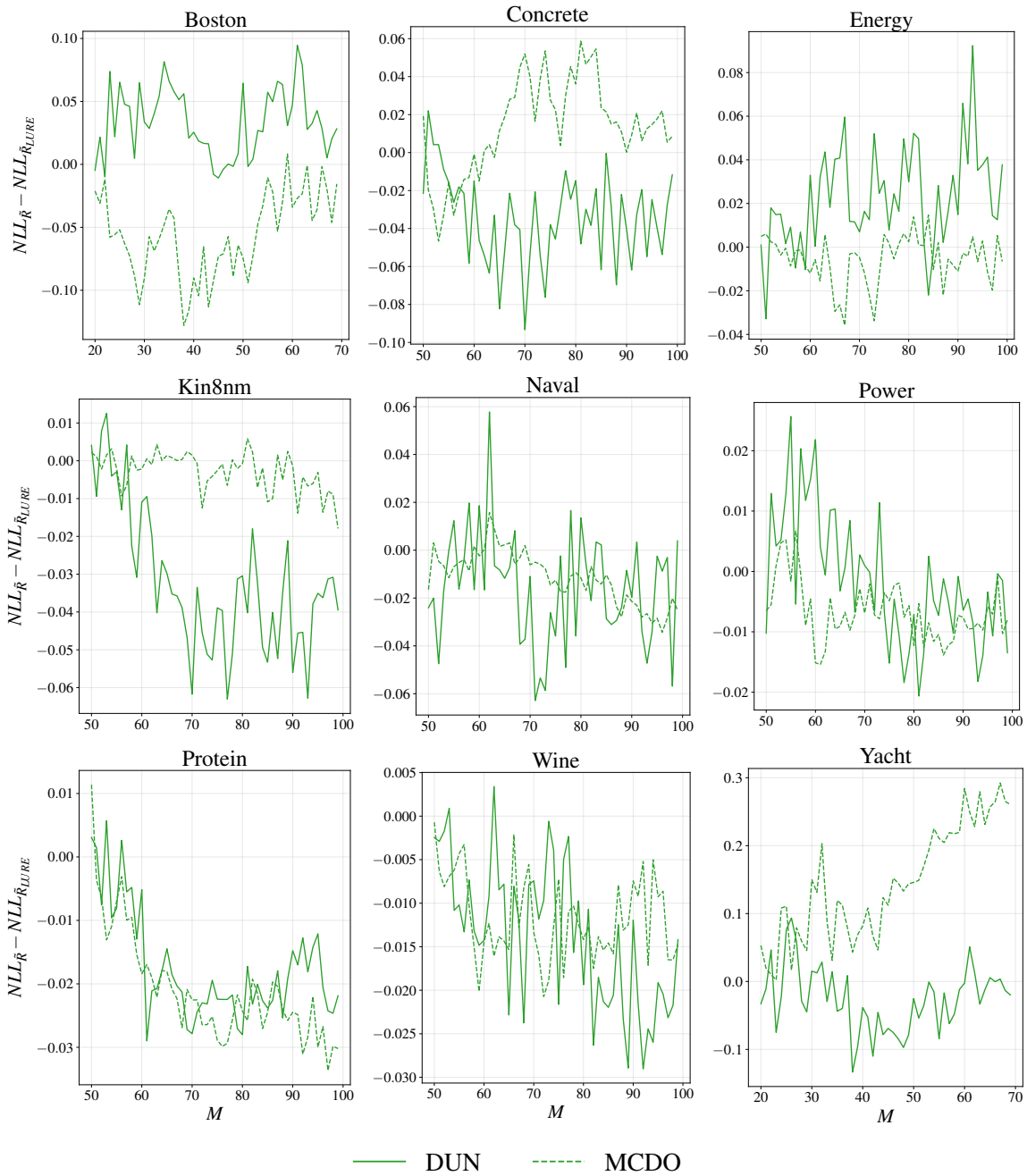


Figure 4.19: Difference in NLL for DUNs (solid lines) and MCDO (dashed lines) trained with  $\tilde{R}$  and trained with  $\tilde{R}_{LURE}$ , evaluated on UCI datasets. A positive value indicates that the weighted estimator  $\tilde{R}_{LURE}$  improves downstream performance, while a negative value indicates that it harms performance.



### 4.4.3 Overfitting bias

The finding that removing active learning bias does not improve downstream performance is consistent with that of Farquhar et al. (2021). They explain this result by considering active learning bias in the context of overall bias, including overfitting bias, which is typically present to varying degrees in overparameterised models whether active learning is used or not. We attempt to isolate the overfitting bias in order to understand how it may interact with active learning bias and influence the results of section 4.4.2. By noting firstly that the parameters  $\theta^*$  optimised for  $\hat{R}$  are generally overfit, and secondly that  $\tilde{R}_{LURE}$  removes active learning bias, we observe that the quantity  $\tilde{R}_{LURE}(\theta^*)$  encompasses only the overfitting bias and the true population risk  $r$ . It is then possible to recover the overfitting bias of a model optimised with either of the discussed risk estimators by subtracting  $\tilde{R}_{LURE}(\theta^*)$  from the true population risk  $r$ :

$$B_{OFB}(\tilde{R}_{(\cdot)}) = r - \tilde{R}_{LURE}(\theta^*) \quad (4.3)$$

where  $\theta^* = \underset{\theta}{\operatorname{argmin}}(\tilde{R}_{(\cdot)})$  and  $\tilde{R}_{(\cdot)}$  is  $\tilde{R}$  or  $\tilde{R}_{LURE}$ . The population risk  $r$  is, as before, approximated by the risk evaluated on the test set  $\mathcal{D}_{\text{test}}$ .

In figure 4.20 we plot the quantities  $B_{OFB}(\tilde{R})$  and  $B_{OFB}(\tilde{R}_{LURE})$  for increasing sizes  $M$  of the labelled training set, for both DUNs and MCDO models. The overfitting bias, in contrast to active learning bias, is positive—overfitting causes the risk as evaluated on the training set to be underestimated, such that the difference between actual and estimated risk is positive. The two sources of bias thus offset each other to a certain extent when considering downstream performance on active learning problems. Comparing the magnitudes of the biases for MCDO in figure 4.17 and figure 4.20, we note that the overfitting bias substantially outweighs the active learning bias, thus explaining why eliminating active learning bias has little observable impact on the final model performance metrics.

The relative magnitudes of active learning bias and overfitting bias are not as consistent for DUNs, with active learning bias in fact exceeding overfitting bias for some datasets (e.g., Boston, Concrete and Power). According to the reasoning presented in Farquhar et al. (2021), removing the active learning bias in cases where  $B_{ALB} > B_{OLB}$  should positively affect performance. It is unclear why this effect is not observed in practice for DUNs. It is possible that the experimental design used by Farquhar et al. (2021) is not directly applicable to the datasets and other specifications used in this work. In particular, we note that the temperature of the proposal distribution used in that work is  $T = 10,000$ , much larger than that used in our experiments ( $T = 10$ ). Since higher temperatures approach a deterministic proposal,

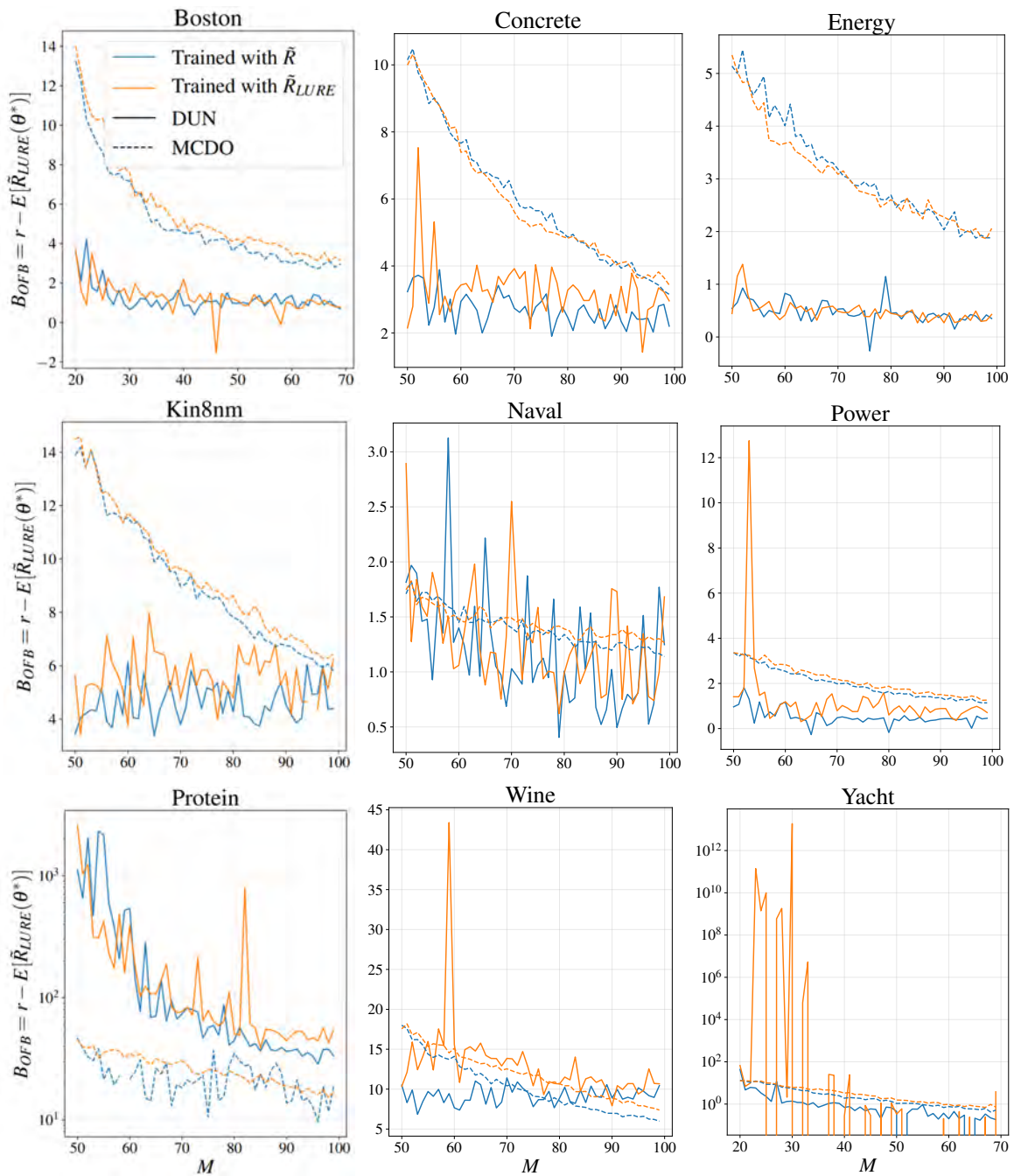


Figure 4.20: Overfitting bias for DUNs (solid lines) and MCDO (dashed lines) trained with  $\tilde{R}$  (in blue) and  $\tilde{R}_{LURE}$  (in orange), evaluated on UCI datasets. Overfitting bias is generally larger in DUNs than for MCDO. NLL for Protein and Yacht is displayed in log scale.

it is possible that this temperature setting interferes with the assumption that  $\alpha(\cdot)$  assigns non-zero probability to all members of  $\mathcal{D}_{\text{train}}$ , and that this in turn affects the generalisability of their findings. We also execute the experiments in this section using  $T = 10,000$ , but find that the results are even less consistent with those of Farquhar et al. (2021) than when using our preferred temperature. Another potential explanation is that there are other interfering sources of bias at play in DUNs that we do not take into account.

Hypothesis 5, outlined in section 3.3, is that DUNs’ flexibility over depth could enable them to avoid or reduce overfitting in low data regimes. The active learning bias correction  $\tilde{R}_{LURE}$  could therefore be beneficial for DUNs, much as it is for inflexible models such as linear regression (as explained in section 2.3). Given the results previously discussed from figure 4.18, it does not appear that the unbiased risk estimator is more useful for DUNs than for other types of neural network. In figure 4.20 we do observe, however, that the magnitude of overfitting bias for DUNs is generally much smaller than that for MCDO (other than the in Protein and Wine datasets). It is thus plausible to conclude that DUNs can, in some cases, be more effective than other forms of BNN in regularising overfitting on small datasets.

#### 4.4.4 Impact of the depth prior

The experiments presented thus far use an uninformative prior for DUNs that assigns uniform probabilities to each subnetwork depth. One of the advantages of DUNs, however, is that it is possible to place informative priors over depth that reflect expectations about the complexity of the model. We propose in hypothesis 2 (section 3.3) that the depth prior in DUNs can be chosen so as to encourage shallower learned networks, which may help to avoid overfitting at the beginning of active learning. With smaller overfitting bias, it might be expected that correcting for active learning bias using  $\tilde{R}_{LURE}$  should yield greater improvements in downstream performance for the informative prior than for the uniform prior.

We evaluate this theory by placing an exponentially decaying prior over depth,

$$\beta_i = \frac{(1 - \gamma)^i}{\sum_{i=0}^D (1 - \gamma)^i} \quad (4.4)$$

with  $\gamma = 0.95$ , and repeating the experiments discussed in section 4.4.2 and section 4.4.3. Figure 4.21 plots the difference in NLL for models trained using  $\tilde{R}$  and  $\tilde{R}_{LURE}$  when a uniform and an exponentially decaying prior are used. The difference is not noticeably different for DUNs with a uniform prior as opposed to a decaying prior. In addition, the difference is close

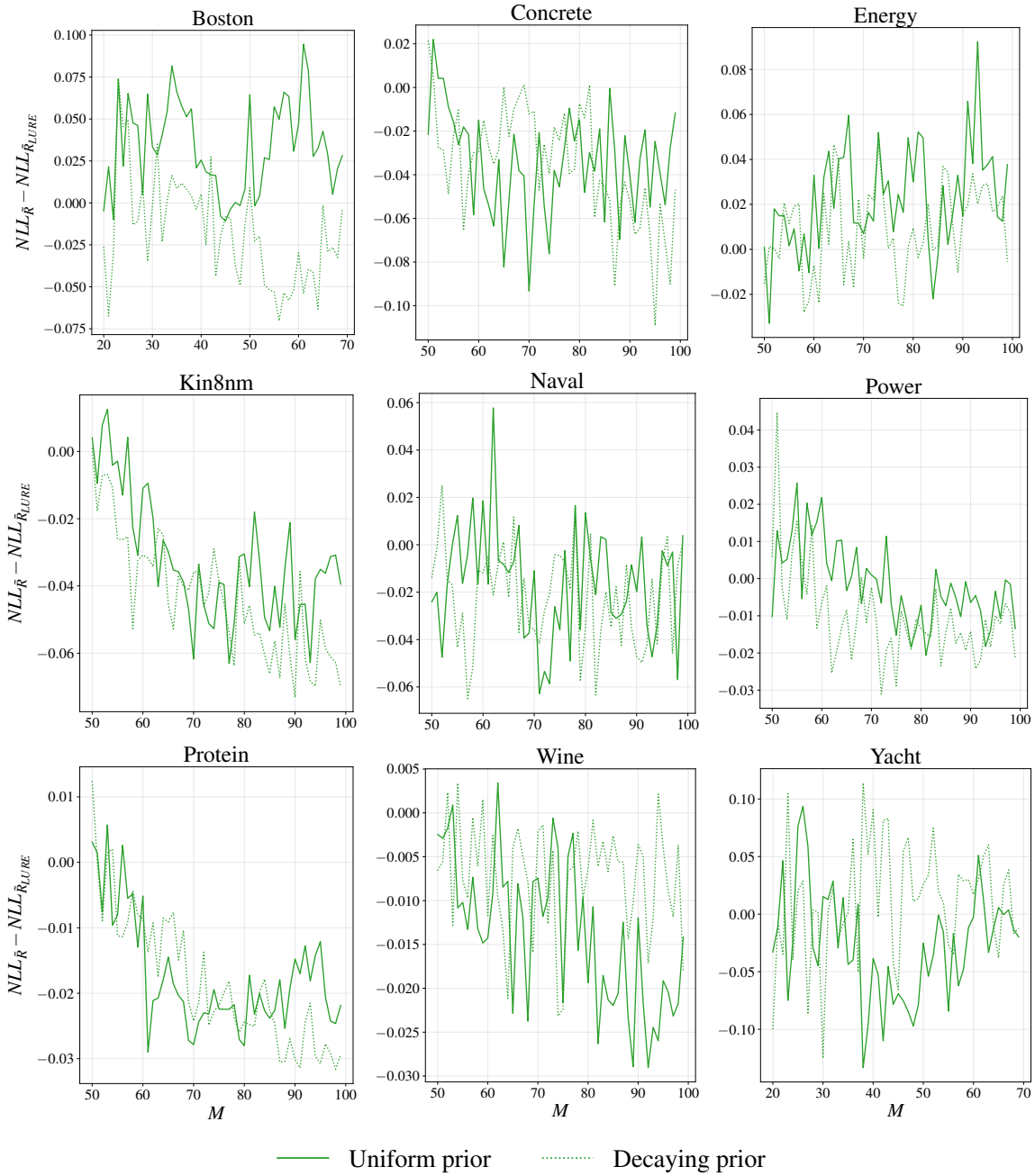


Figure 4.21: Difference in NLL for DUNs with a uniform prior (solid lines) and decaying prior (dotted lines) trained with  $\tilde{R}$  and trained with  $\tilde{R}_{LURE}$ , evaluated on UCI datasets. A positive value indicates that the weighted estimator  $\tilde{R}_{LURE}$  improves downstream performance, while a negative value indicates that it harms performance.

to zero for the majority of the datasets and in some cases slightly negative, indicating that the unbiased estimator harms performance. The hypothesis that the unbiased risk estimator should benefit DUNs with a decreasing prior appears, therefore, not to hold.

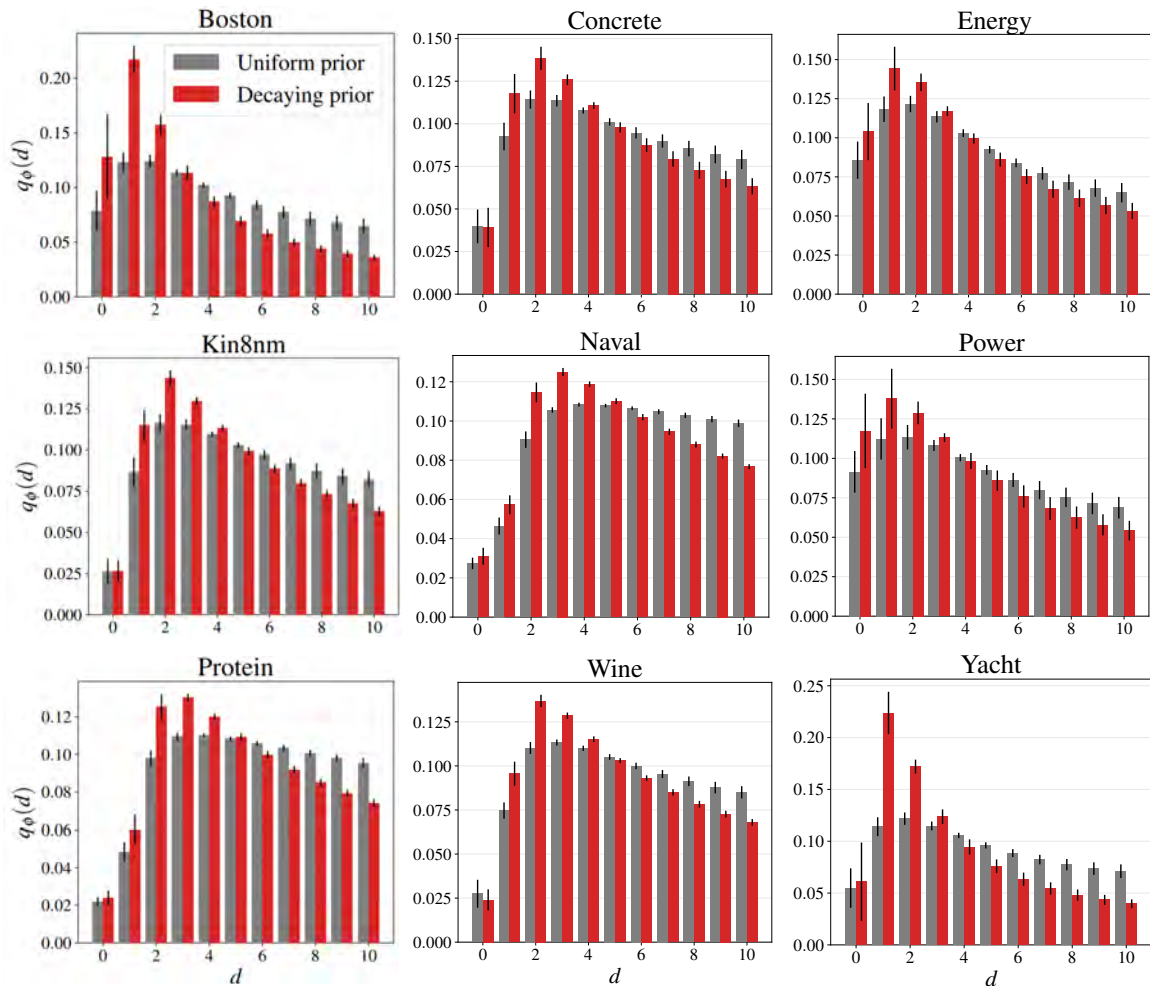


Figure 4.22: Posterior probabilities over depth for DUNs after the first acquisition step, using a uniform prior over depth (grey bars) and an exponentially decaying prior (red bars).

To investigate why this may be the case, we first confirm that the decreasing prior does influence the depth posterior as expected. As shown in figure 4.22 for the initial dataset used in active learning, when the prior probabilities decrease with depth, the resulting posterior places more mass on shallower layers than the posterior with uniform prior. The difference in posterior mass does not, however, appear to have a strong enough regularising effect on the learned function to influence overall performance—in figure 4.23, the NLL for DUNs with uniform and decaying priors is almost identical. It is possible that as the train set grows

in size during active learning, the influence of the prior observed in figure 4.22 quickly becomes negligible. Alternatively, given that the magnitude of overfitting bias is already small in DUNs with a uniform prior (as shown in figure 4.20, relative to MCDO), it is possible that attempting to further reduce this source of bias is both difficult and has limited impact.

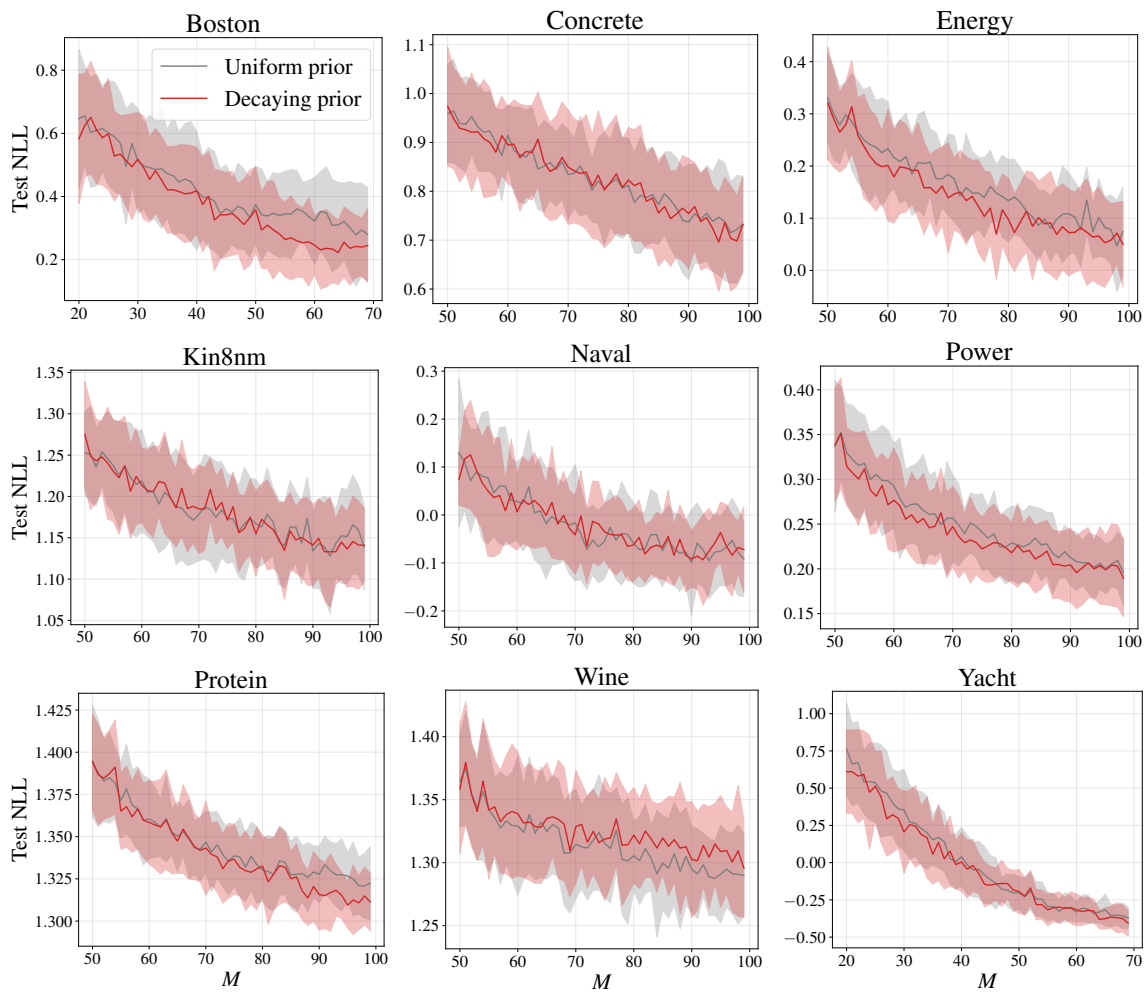


Figure 4.23: Test NLL vs. number of training points for DUNs evaluated on UCI datasets. A uniform depth prior is compared to a decaying prior.

Quantifying the overfitting bias confirms that the decaying prior does little to reduce this source of bias. Figure 4.24 replicates figure 4.20 but for DUNs with a uniform and a decaying prior, and shows that the magnitude of overfitting bias is similar for both priors. That the decaying prior is ineffective in moderating the overfitting bias and improving downstream model performance explains why we do not observe any benefit from combining the unbiased risk estimator  $\tilde{R}_{LURE}$  with a decaying prior. These findings support the claim in section 2.2.3

---

that a disadvantage of DUNs is that the prior has limited impact on the loss function and the posterior. Since with DUNs the KL term in equation (2.36) is computed between distributions over depth, which have far fewer parameters than distributions over the network weights, the scale of the KL term is comparatively smaller in the DUN loss function than it would be in the BNN loss function. The prior over depth thus has a more limited regularising effect than a prior over weights.

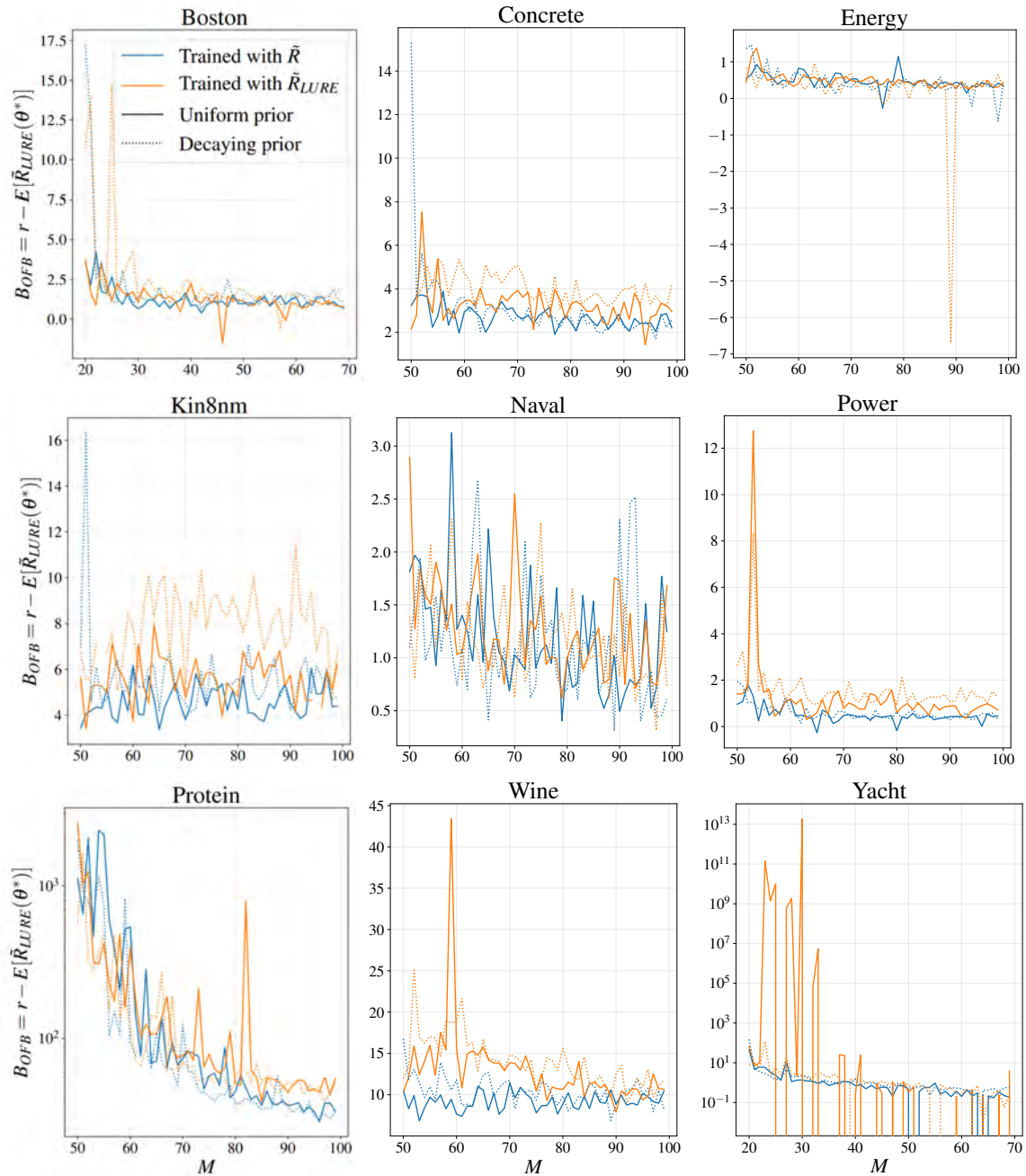


Figure 4.24: Overfitting bias for DUNs with a uniform prior (solid lines) and decaying prior (dotted lines) trained with  $\tilde{R}$  (in blue) and  $\tilde{R}_{LURE}$  (in orange), evaluated on UCI datasets. Overfitting bias is similar in magnitude for the uniform prior and decaying prior. NLL for Protein and Yacht is displayed in log scale.



# 5

## Conclusion

This final chapter is dedicated to discussing the results obtained in this thesis and promising avenues for future work.

### 5.1 Summary of findings

The objective of this thesis is to explore the application of a recently proposed BNN variant, depth uncertainty networks, to active learning problems. The overarching motivation for this investigation is that particular properties of DUNs—namely, flexibility over network depth and the resulting adaptability of network complexity to the given dataset, along with the ability to exactly compute model uncertainty estimates—are expected to be advantageous in addressing certain challenges of active learning. We propose and empirically test the following hypotheses:

**1. The ability to infer depth is a useful property for active learning that enables DUNs to adapt the complexity of the model to the size of the training dataset.**

In section 4.3 we show that this hypothesis holds, by demonstrating that the optimal depth inferred by DUNs increases as more labelled data are acquired. Additionally, we show that DUNs outperform MCDO and MFVI in most of the cases tested; however, this finding could also be attributed to other strengths of DUNs, such as the quality of their uncertainty estimates, or the fact that approximations are not required to compute  $\alpha_{\text{BALD}}$ .

**2. The prior over depth can be leveraged as a regularisation mechanism to further reduce overfitting when training sets are small.**

By comparing the performance of DUNs with a uniform prior and exponentially decaying prior in section 4.4.4, we show that the prior has limited influence on the posterior and the extent of overfitting. This is identified as a limitation of DUNs.

**3. Truncating BALD scores to the value of the maximum expected predictive variance helps to counteract the effects of model misspecification bias.**

An illustrative example provided in section 4.2.2 shows that truncation successfully minimises clustering in high-variance regions of the input space. This method appears only to be relevant in the initial stages of acquisition and does not materially impact overall performance.

**4. Introducing stochasticity to information-based selection improves the diversity of acquired examples and helps to address loss of information due to correlation.**

The stochastic modification of the BALD acquisition function is shown in section 4.2.3 to recover some of the performance loss caused by acquiring correlated batches, however significant gains over random acquisition are not achieved.

**5. Overfitting bias is small or non-existent in DUNs with well-chosen priors over depth. As a result, removing active learning bias improves downstream performance.**

By applying the approach of Farquhar et al. (2021) to quantifying biases in active learning, we confirm that overfitting bias is smaller for DUNs than for MCDO, in section 4.4. Despite active learning bias being larger in magnitude than overfitting bias in several cases for DUNs, however, correcting for active learning bias does not impact downstream performance, a finding that is inconsistent with the results of Farquhar et al. (2021).

## 5.2 Limitations and future work

### Bias results verification

The finding that correcting for active learning bias in DUNs does not lead to performance improvements, despite the fact that this source of bias outweighs overfitting bias for several of the datasets, is unexpected and dissatisfying, particularly given that the magnitude of overfitting is smaller for DUNs than for other BNN inference methods. Further work could seek to find theoretically grounded explanations for this result. An initial avenue of investigation could be to verify the results found in this work on the unbalanced MNIST dataset and FashionMNIST dataset used in Farquhar et al. (2021).

### Complex data

The experiments conducted in this thesis are limited to simple, low-dimensional datasets. Active learning, in contrast, is designed for situations in which label generation is expensive, a setting that one might expect to be associated with complex and high-dimensional data such

as medical images. It is also plausible to assume that active learning should deliver more value relative to passive learning for complex datasets with larger input spaces, or for “messy” data that contain greater diversity in the informativeness of individual examples (e.g., if the dataset contains duplicate or ambiguous examples). A straightforward extension of this work is to apply the experiments to more complex datasets, for which more meaningful differences in performance between active and passive acquisition, and between the inference methods tested, may be observed.

### **Improved DUNs**

Motivated by the finding that an exponentially decaying prior has limited impact on downstream performance, an avenue for further research is the development of improved DUNs in which the prior over depth plays a larger role in controlling model complexity. One approach is to use the prior for each depth as a hyperprior for the prior precision of the corresponding layer in the neural network. Estimates of the depth hyperparameters then directly affect the model weights. The hyperprior over depth is expected to have a stronger influence on the model complexity because the KL divergence between the weight-space prior and approximate posterior will be larger than the KL divergence between the depth prior and approximate posterior, due to the large size of the weight space.

### **Batch acquisition**

We employ batch acquisition, but do not implement fully batch-aware acquisition strategies such as BatchBALD. This decision is made in the interests of computational efficiency and because the primary focus of this work is the study of biases and the performance of DUNs in active learning, not the analysis of acquisition functions. Further work could seek to verify that our conclusions also hold for more sophisticated batch-aware acquisition functions and for the serial acquisition case.



# References

- Antorán, J. (2019). Understanding uncertainty in Bayesian neural networks. *MPhil thesis, University of Cambridge*.
- Antorán, J., Allingham, J., and Hernández-Lobato, J. M. (2020). Depth uncertainty in neural networks. *Advances in Neural Information Processing Systems*, 33.
- Antorán, J., Allingham, J. U., and Hernández-Lobato, J. M. (2020). Variational depth search in ResNets. *CoRR*, abs/2002.02797.
- Ash, J. T., Zhang, C., Krishnamurthy, A., Langford, J., and Agarwal, A. (2020). Deep batch active learning by diverse, uncertain gradient lower bounds. In *International Conference on Learning Representations*.
- Beirlant, J., Dudewicz, E. J., Györfi, L., Van der Meulen, E. C., et al. (1997). Nonparametric entropy estimation: An overview. *International Journal of Mathematical and Statistical Sciences*, 6(1):17–39.
- Beygelzimer, A., Dasgupta, S., and Langford, J. (2009). Importance weighted active learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 49–56.
- Bhatt, U., Antorán, J., Zhang, Y., Liao, Q. V., Sattigeri, P., Fogliato, R., Melançon, G., Krishnan, R., Stanley, J., Tickoo, O., et al. (2021). Uncertainty as a form of transparency: Measuring, communicating, and using uncertainty. In *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*, pages 401–413.
- Bishop, C. M. (2007). *Pattern Recognition and Machine Learning, 5th Edition*. Information science and statistics. Springer.
- Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. (2017). Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877.
- Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. (2015). Weight uncertainty in neural networks. In *International Conference on Machine Learning*, pages 1613–1622. PMLR.
- Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language models are few-shot learners. In

- Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.
- Chu, W., Zinkevich, M., Li, L., Thomas, A., and Tseng, B. (2011). Unbiased online active learning in data streams. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 195–203. Association for Computing Machinery.
- Cohn, D. A., Ghahramani, Z., and Jordan, M. I. (1995). Active learning with statistical models. Technical report, MASSACHUSETTS INST OF TECH CAMBRIDGE ARTIFICIAL INTELLIGENCE LAB.
- Cortes, C., DeSalvo, G., Mohri, M., Zhang, N., and Gentile, C. (2019). Active learning with disagreement graphs. In *International Conference on Machine Learning*, pages 1379–1387. PMLR.
- Dasgupta, S. and Hsu, D. (2008). Hierarchical sampling for active learning. In *Proceedings of the 25th International Conference on Machine Learning*, pages 208–215.
- Depeweg, S., Hernandez-Lobato, J.-M., Doshi-Velez, F., and Udluft, S. (2018). Decomposition of uncertainty in Bayesian deep learning for efficient and risk-sensitive learning. In *International Conference on Machine Learning*, pages 1184–1193. PMLR.
- D’Amour, A., Heller, K., Moldovan, D., Adlam, B., Alipanahi, B., Beutel, A., Chen, C., Deaton, J., Eisenstein, J., Hoffman, M., et al. (2020). Underspecification presents challenges for credibility in modern machine learning. arxiv 2020. *arXiv preprint arXiv:2011.03395*.
- Farquhar, S., Gal, Y., and Rainforth, T. (2021). On statistical bias in active learning: How and when to fix it. *International Conference on Learning Representations*.
- Foong, A., Burt, D., Li, Y., and Turner, R. (2020). On the expressiveness of approximate inference in bayesian neural networks. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 15897–15908. Curran Associates, Inc.
- Foong, A. Y. K., Li, Y., Hernández-Lobato, J. M., and Turner, R. E. (2019). ‘In-between’ uncertainty in Bayesian neural networks. *CoRR*, abs/1906.11537.
- Gal, Y. (2016). Uncertainty in deep learning. *PhD thesis, University of Cambridge*.
- Gal, Y. and Ghahramani, Z. (2016). Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, pages 1050–1059. PMLR.
- Gal, Y., Islam, R., and Ghahramani, Z. (2017). Deep Bayesian active learning with image data. In *International Conference on Machine Learning*, pages 1183–1192. PMLR.
- Ganti, R. and Gray, A. (2012). UPAL: Unbiased pool based active learning. In Lawrence, N. D. and Girolami, M., editors, *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics*, volume 22 of *Proceedings of Machine Learning Research*, pages 422–431. PMLR.

- Graves, A. (2011). Practical variational inference for neural networks. In *Advances in neural information processing systems*, pages 2348–2356. Citeseer.
- He, K., Zhang, X., Ren, S., and Sun, J. (2016). Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer.
- Hernández-Lobato, J. M. and Adams, R. (2015). Probabilistic backpropagation for scalable learning of Bayesian neural networks. In *International Conference on Machine Learning*, pages 1861–1869. PMLR.
- Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580.
- Hinton, G. E. and Van Camp, D. (1993). Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the sixth annual conference on Computational learning theory*, pages 5–13.
- Houlsby, N., Huszar, F., Ghahramani, Z., and Lengyel, M. (2011). Bayesian active learning for classification and preference learning. *CoRR*, abs/1112.5745.
- Huang, S.-J., Zhao, J.-W., and Liu, Z.-Y. (2018). Cost-effective training of deep cnns with active model adaptation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1580–1588.
- Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.
- Izmailov, P., Maddox, W. J., Kirichenko, P., Garipov, T., Vetrov, D., and Wilson, A. G. (2020). Subspace inference for Bayesian deep learning. In *Uncertainty in Artificial Intelligence*, pages 1169–1179. PMLR.
- Jing, L. and Tian, Y. (2020). Self-supervised visual feature learning with deep neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*.
- Kingma, D. P., Salimans, T., and Welling, M. (2015). Variational dropout and the local reparameterization trick. In Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.
- Kingma, D. P. and Welling, M. (2014). Auto-encoding variational Bayes. In *Proceedings of the 2nd International Conference on Learning Representations*.
- Kirsch, A., Van Amersfoort, J., and Gal, Y. (2019). BatchBALD: Efficient and diverse batch acquisition for deep Bayesian active learning. *Advances in neural information processing systems*, 32:7026–7037.
- Kozachenko, L. and Leonenko, N. N. (1987). Sample estimate of the entropy of a random vector. *Problemy Peredachi Informatsii*, 23(2):9–16.
- Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105.

- Kullback, S. and Leibler, R. A. (1951). On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86.
- Lakshminarayanan, B., Pritzel, A., and Blundell, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.
- LeCun, Y., Cortes, C., and Burges, C. (1998). MNIST handwritten digit database, 1998. URL <http://www.research.att.com/~yann/ocr/mnist>.
- Lewis, D. D. and Gale, W. A. (1994). A sequential algorithm for training text classifiers. In *SIGIR'94*, pages 3–12. Springer.
- MacKay, D. J. (1992a). Information-based objective functions for active data selection. *Neural computation*, 4(4):590–604.
- MacKay, D. J. (1992b). A practical Bayesian framework for backpropagation networks. *Neural computation*, 4(3):448–472.
- MacKay, D. J. (2003). *Information theory, inference and learning algorithms*. Cambridge University Press.
- Mohamed, S., Rosca, M., Figurnov, M., and Mnih, A. (2020). Monte carlo gradient estimation in machine learning. *Journal of Machine Learning Research*, 21(132):1–62.
- Neal, R. M. (1995). *Bayesian Learning for Neural Networks*. dissertation, University of Toronto.
- Osband, I. (2016). Risk versus uncertainty in deep learning: Bayes, bootstrap and the dangers of dropout. In *NIPS Workshop on Bayesian Deep Learning*, volume 192.
- Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. (2019). Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32:8026–8037.
- Pinsler, R., Gordon, J., Nalisnick, E., and Hernández-Lobato, J. M. (2019). Bayesian batch active learning as sparse subset approximation. *Advances in neural information processing systems*, 32:6359–6370.
- Robbins, H. and Monro, S. (1951). A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407.
- Sener, O. and Savarese, S. (2018). Active learning for convolutional neural networks: A core-set approach. In *International Conference on Learning Representations*.
- Settles, B. (2010). Active learning literature survey. *Machine Learning*, 15(2):201–221.
- Settles, B. (2011). From theories to queries: Active learning in practice. In *Active Learning and Experimental Design workshop In conjunction with AISTATS 2010*, pages 1–18. JMLR Workshop and Conference Proceedings.



- Shannon, C. E. (1948). A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423.
- Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958.
- Tong, S. and Koller, D. (2001). Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research*, 2(Nov):45–66.
- Trippe, B. and Turner, R. (2018). Overpruning in variational Bayesian neural networks. *CoRR*, abs/1801.06230.
- Verdoja, F. and Kyrki, V. (2020). Notes on the behavior of MC dropout. *arXiv preprint arXiv:2008.02627*.
- Wainwright, M. J. and Jordan, M. I. (2008). Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1–2):1–305.
- Zeiler, M. D. and Fergus, R. (2014). Visualizing and understanding convolutional networks. In Fleet, D. J., Pajdla, T., Schiele, B., and Tuytelaars, T., editors, *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I*, volume 8689 of *Lecture Notes in Computer Science*, pages 818–833. Springer.



# Appendix A

## Additional results

### A.1 Acquisition function comparisons

We provide additional results for the comparison of acquisition strategies, including evaluations on the toy datasets and in terms of RMSE for the UCI datasets. Main results are presented in section 4.2.

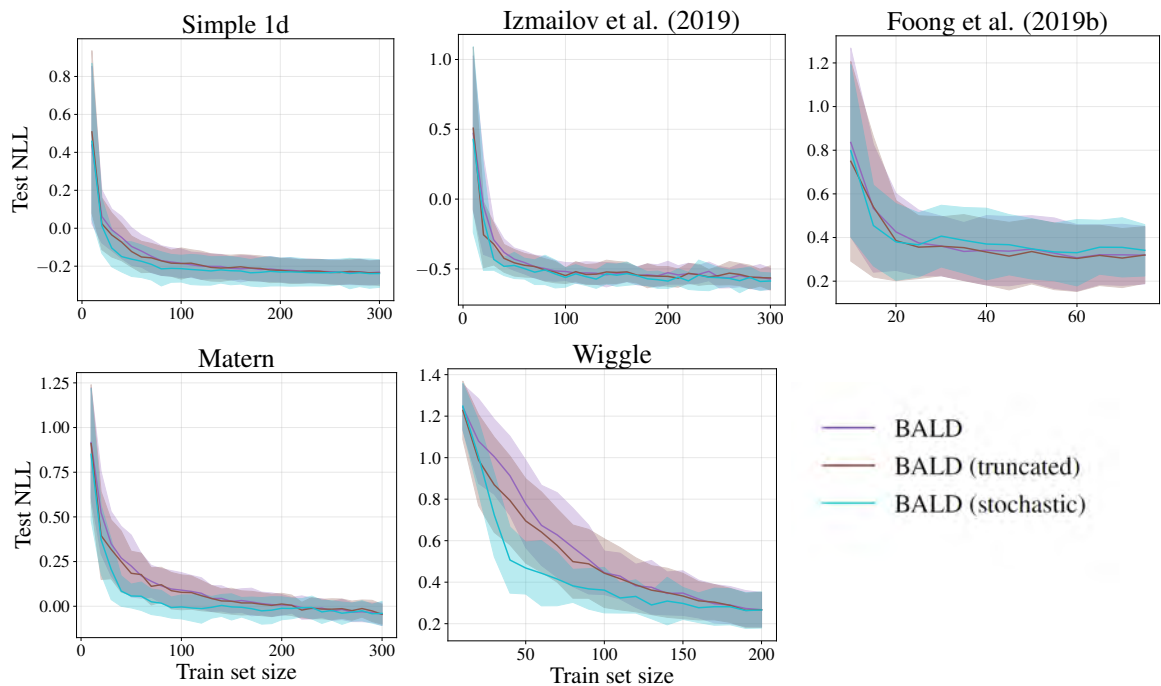


Figure A.1: NLL vs. number of training points for DUNs evaluated on toy datasets. Truncated BALD and stochastic BALD acquisition functions are compared to standard BALD.

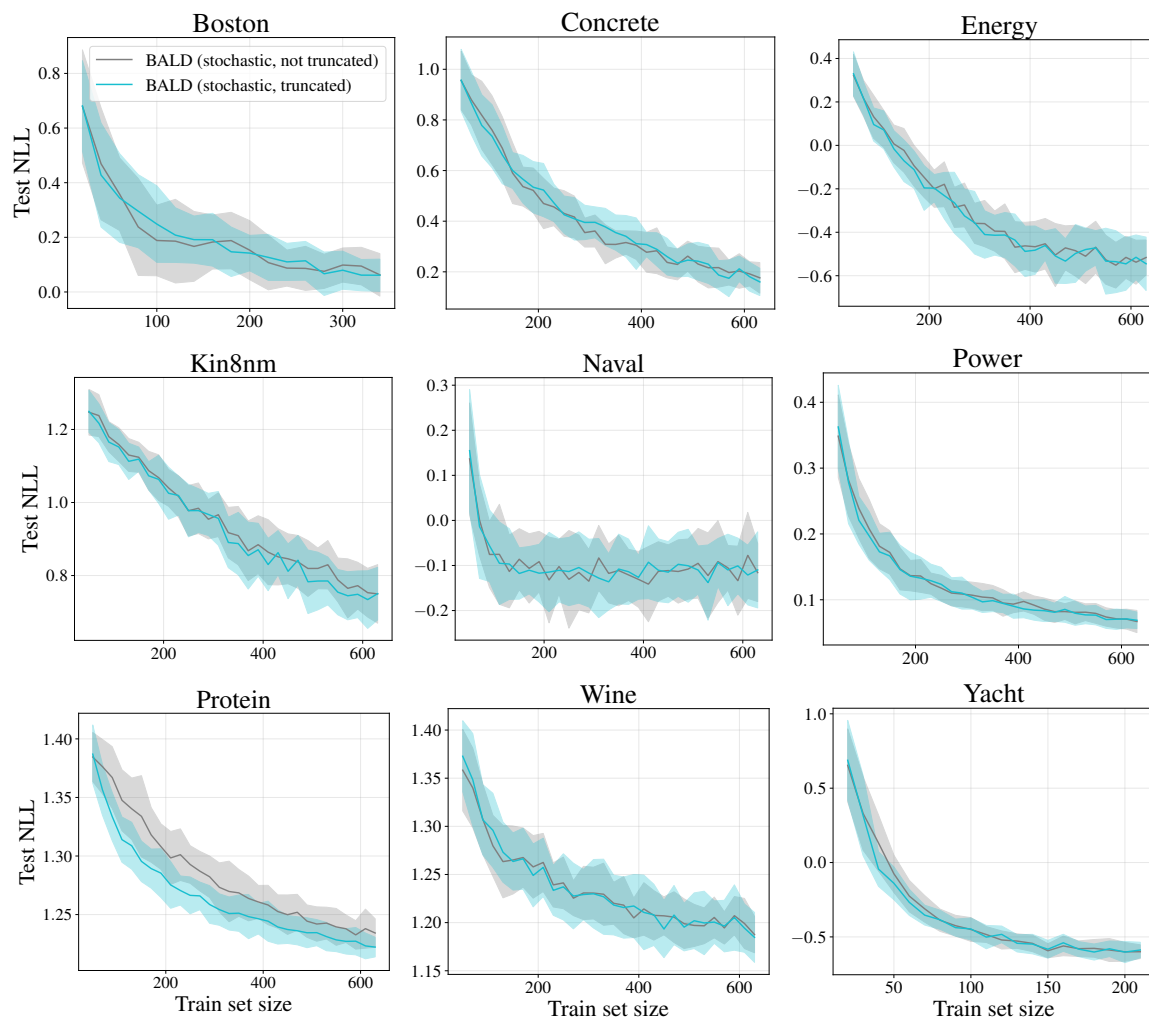


Figure A.2: NLL vs. number of training points for DUNs evaluated on toy datasets. Stochastic BALD acquisition with and without prior truncation are compared.

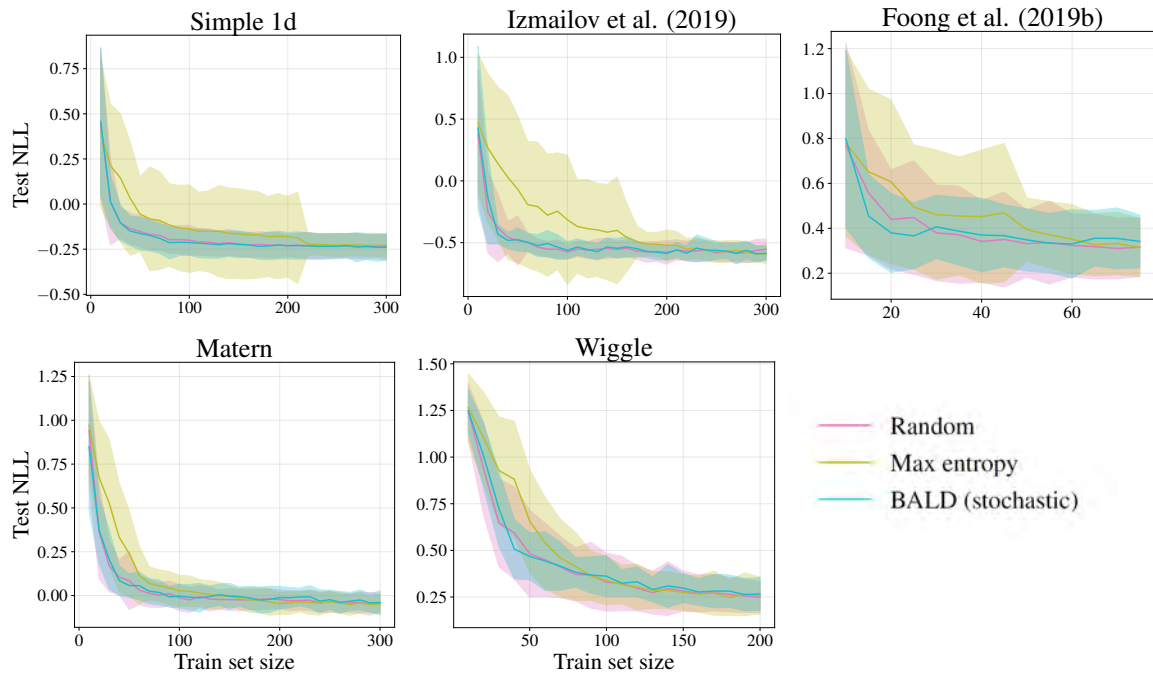


Figure A.3: NLL vs. number of training points for DUNs evaluated on toy datasets. Maximum entropy and stochastic BALD are compared to a random acquisition baseline.

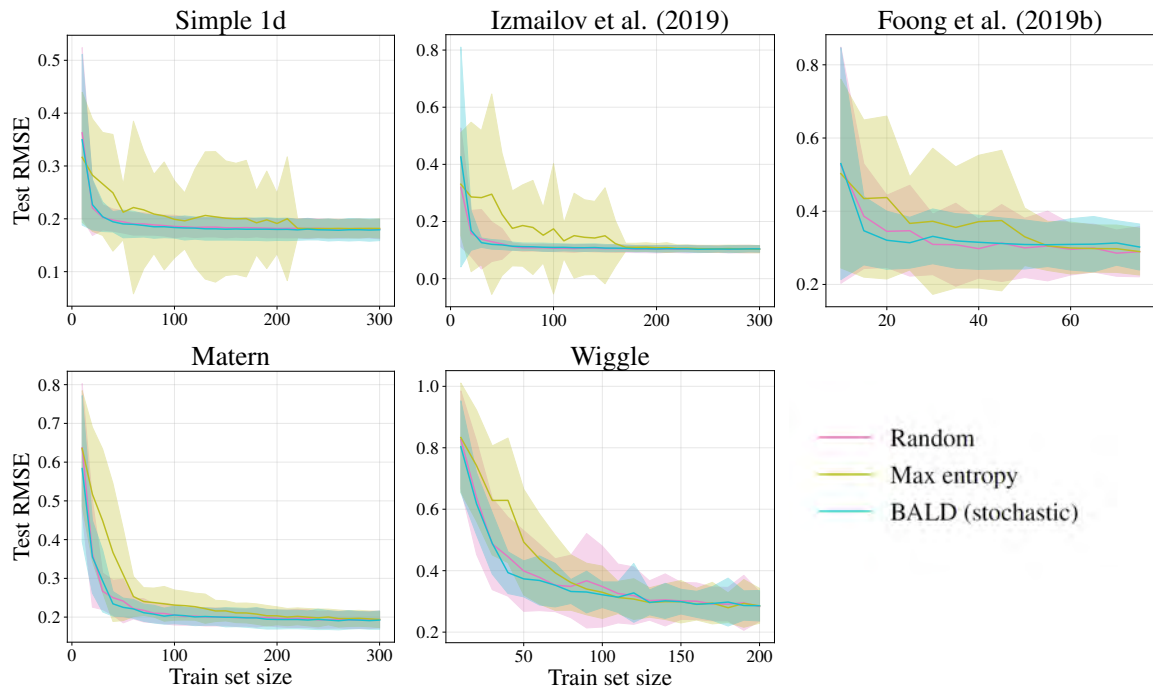


Figure A.4: RMSE vs. number of training points for DUNs evaluated on toy datasets. Maximum entropy and stochastic BALD are compared to a random acquisition baseline.

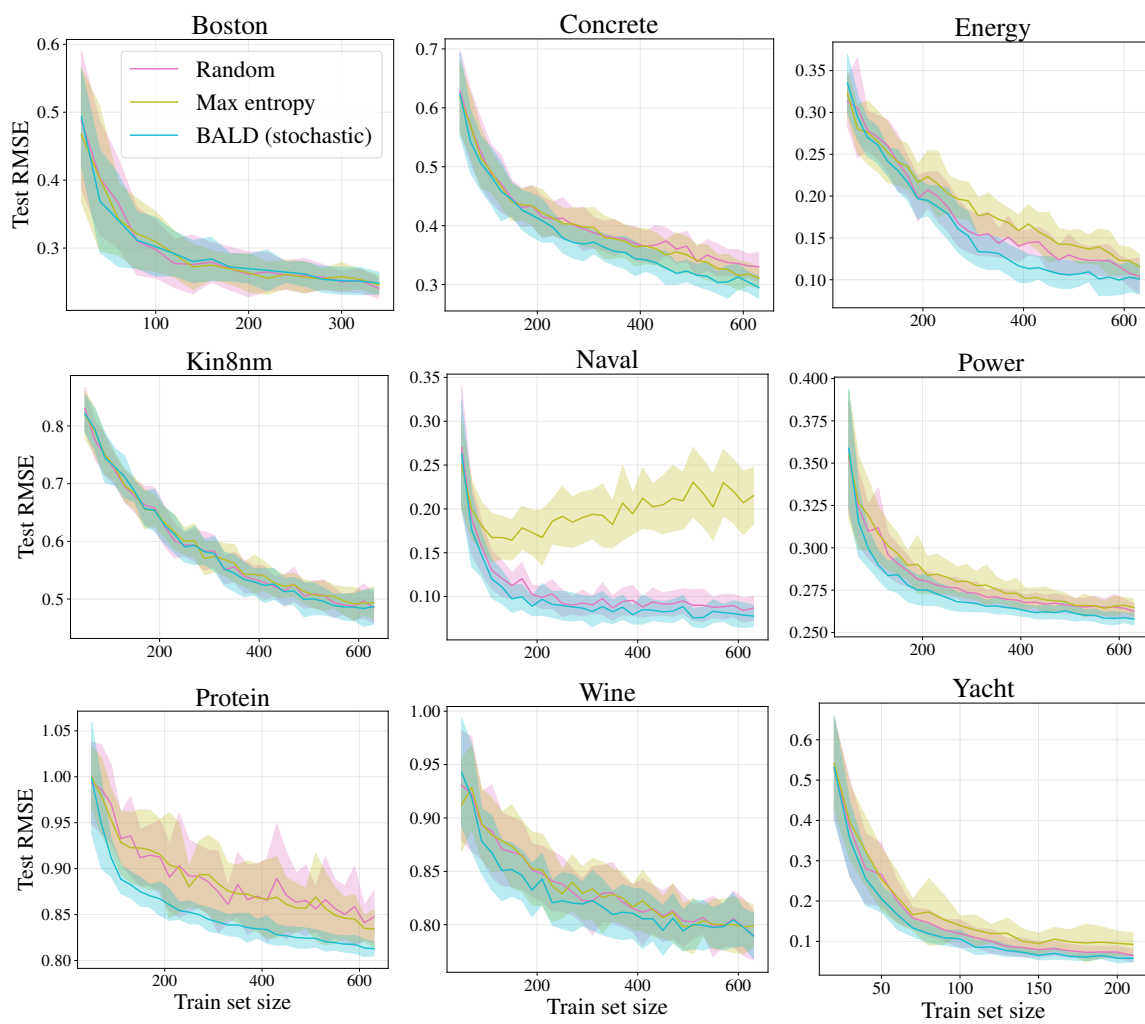


Figure A.5: RMSE vs. number of training points for DUNs evaluated on UCI datasets. Maximum entropy and stochastic BALD acquisition functions are compared to a random acquisition baseline.

## A.2 DUN and baselines comparisons

We provide results for the comparison of DUNs, MCDO and MFVI in terms of RMSE. Figure A.6 is the equivalent of figure 4.9 in terms of RMSE instead of NLL, while figure A.7 is the equivalent of figure 4.13 in terms of RMSE.

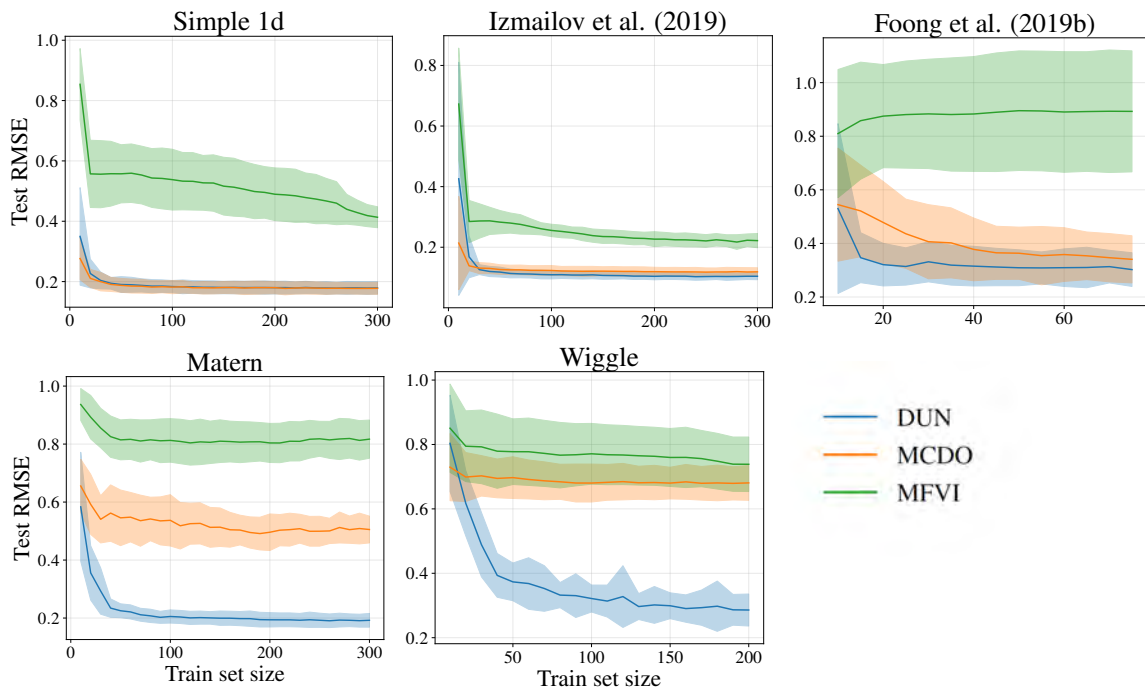


Figure A.6: RMSE vs. number of training points using stochastic BALD acquisition function evaluated on toy datasets. The performance of DUNs, MCDO and MFVI is compared.

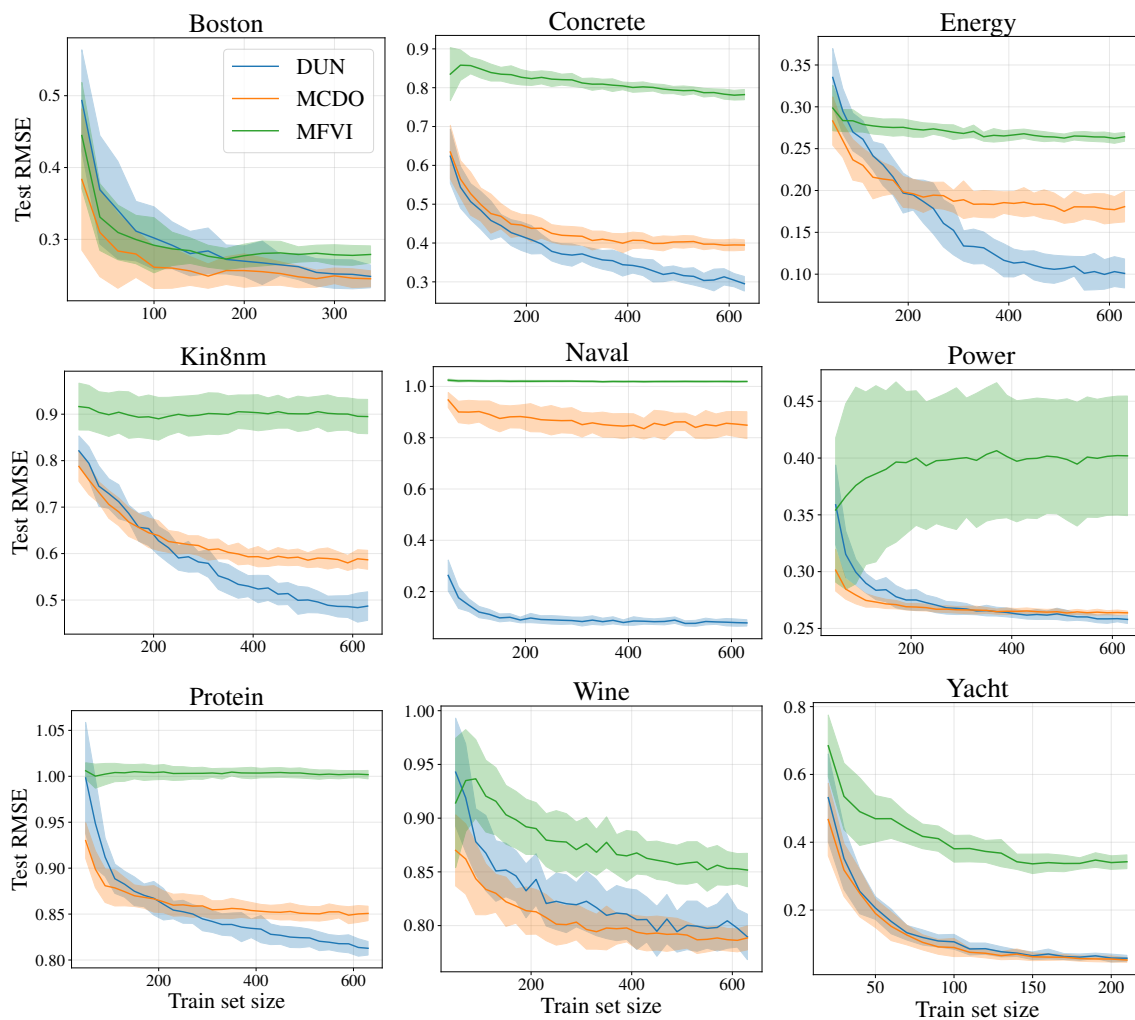


Figure A.7: RMSE vs. number of training points using stochastic BALD acquisition function evaluated on UCI datasets. The performance of DUNs, MCDO and MFVI is compared.



### A.3 Active learning bias experiments

We provide additional results for the investigation of unbiased risk estimators in section 4.4.2. Figure A.8 is equivalent to figure 4.18, which compares downstream performance with the biased and unbiased risk estimators, in terms of RMSE instead of NLL. Figure A.9 also replicates this comparison, for a DUN with exponentially decaying instead of uniform prior.

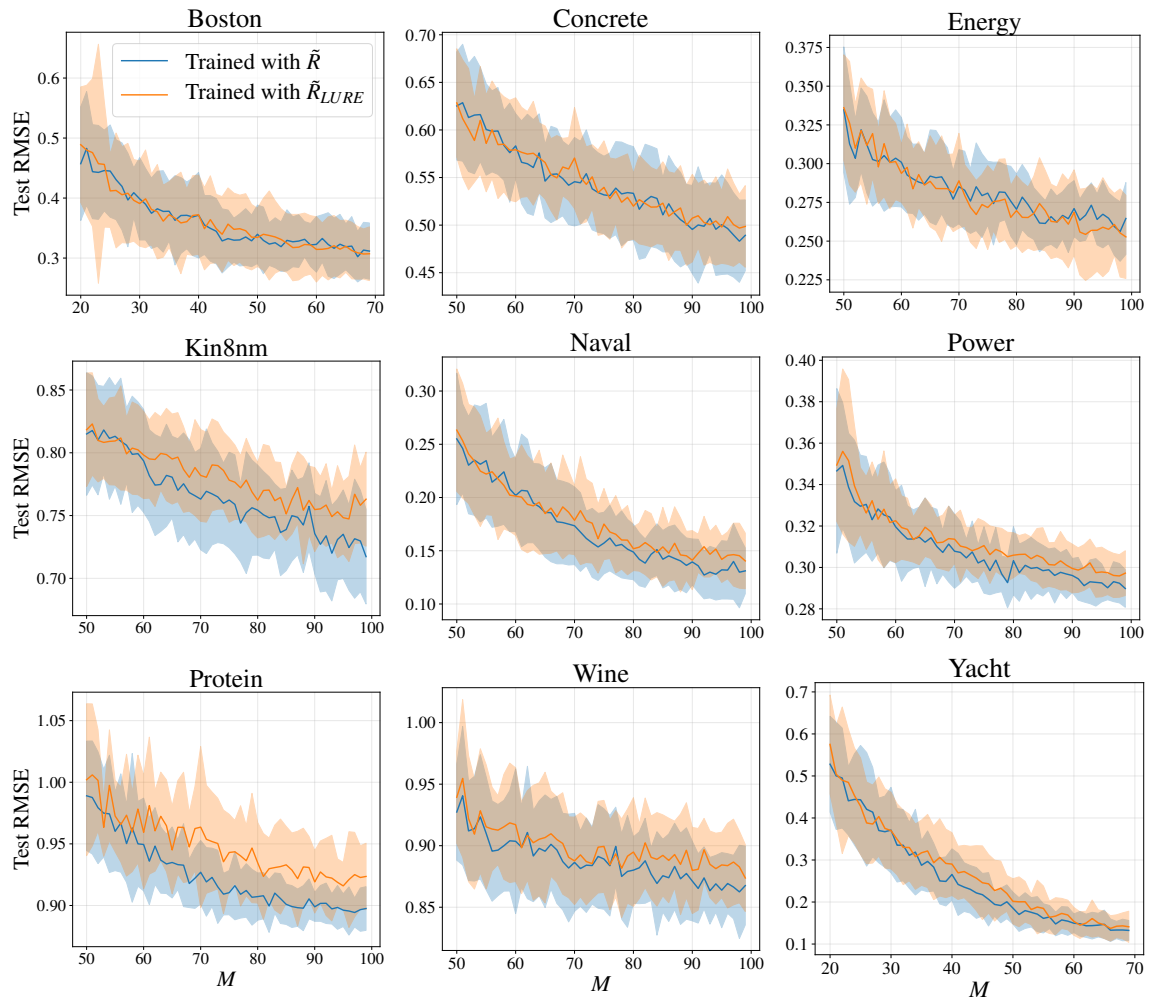


Figure A.8: RMSE for DUNs trained with  $\tilde{R}$  (in blue) and  $\tilde{R}_{LURE}$  (in orange) evaluated on UCI datasets. A larger value of the orange line indicates that training with  $\tilde{R}_{LURE}$  harms performance, despite removing active learning bias.

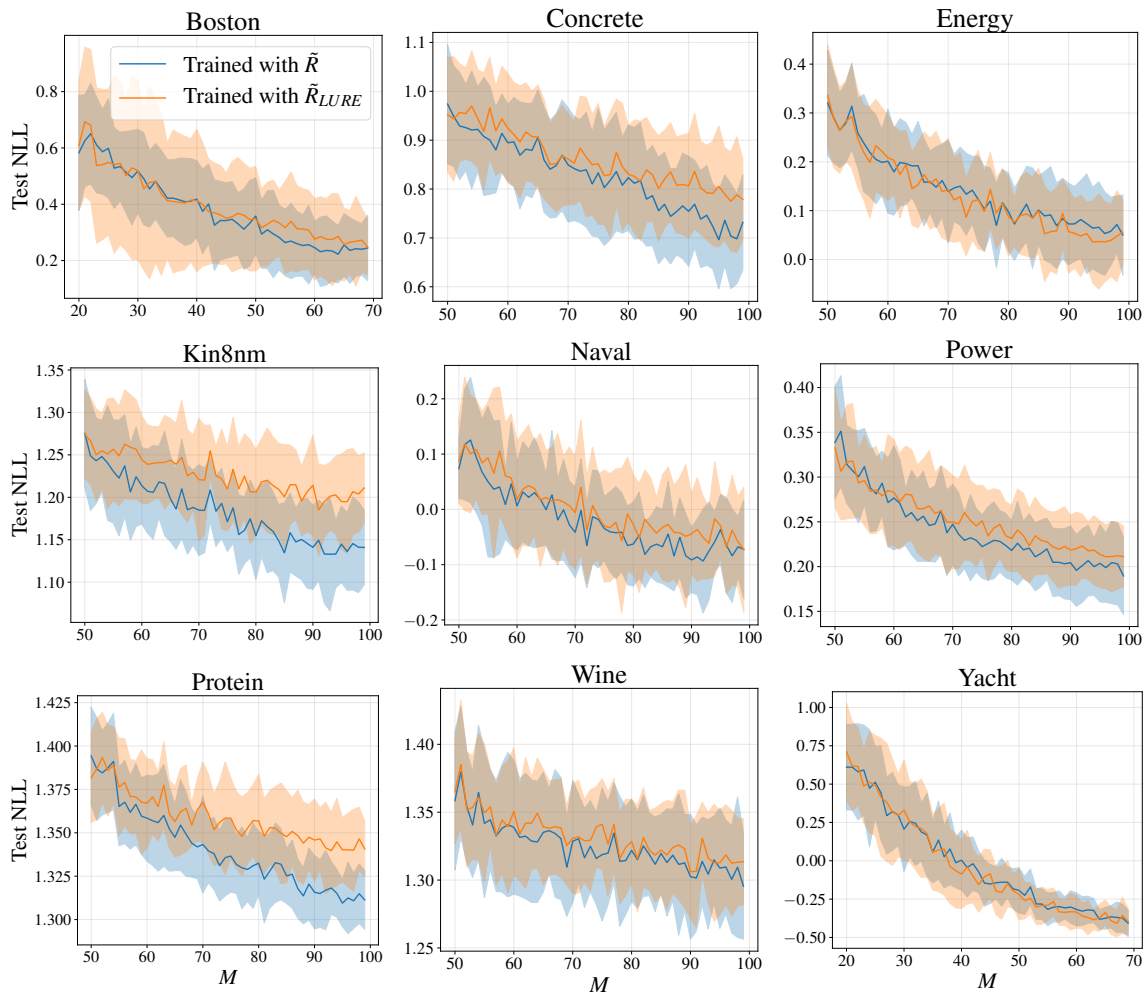
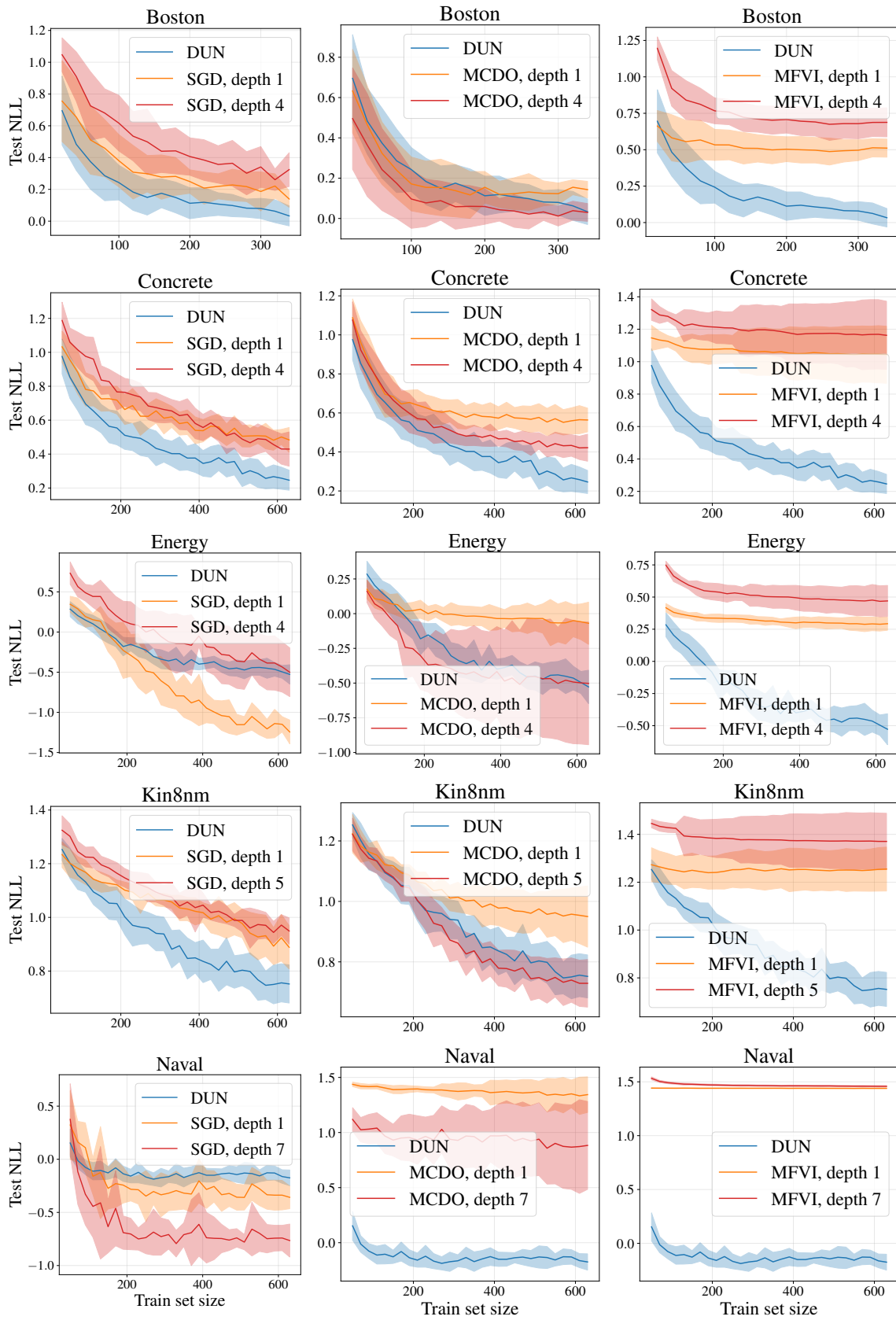


Figure A.9: NLL for DUNs with decaying priors trained with  $\tilde{R}$  (in blue) and  $\tilde{R}_{LURE}$  (in orange) evaluated on UCI datasets. A larger value of the orange line indicates that training with  $\tilde{R}_{LURE}$  harms performance, despite removing active learning bias.

## **A.4 Alternative depths for baseline methods**

The following plots compare the performance of DUNs to the baseline methods (in addition to SGD) with different depth networks for the baseline methods. For each dataset and baseline method, a single hidden layer network and a network with depth equal to the highest posterior probability depth found by the DUN for that dataset are shown.



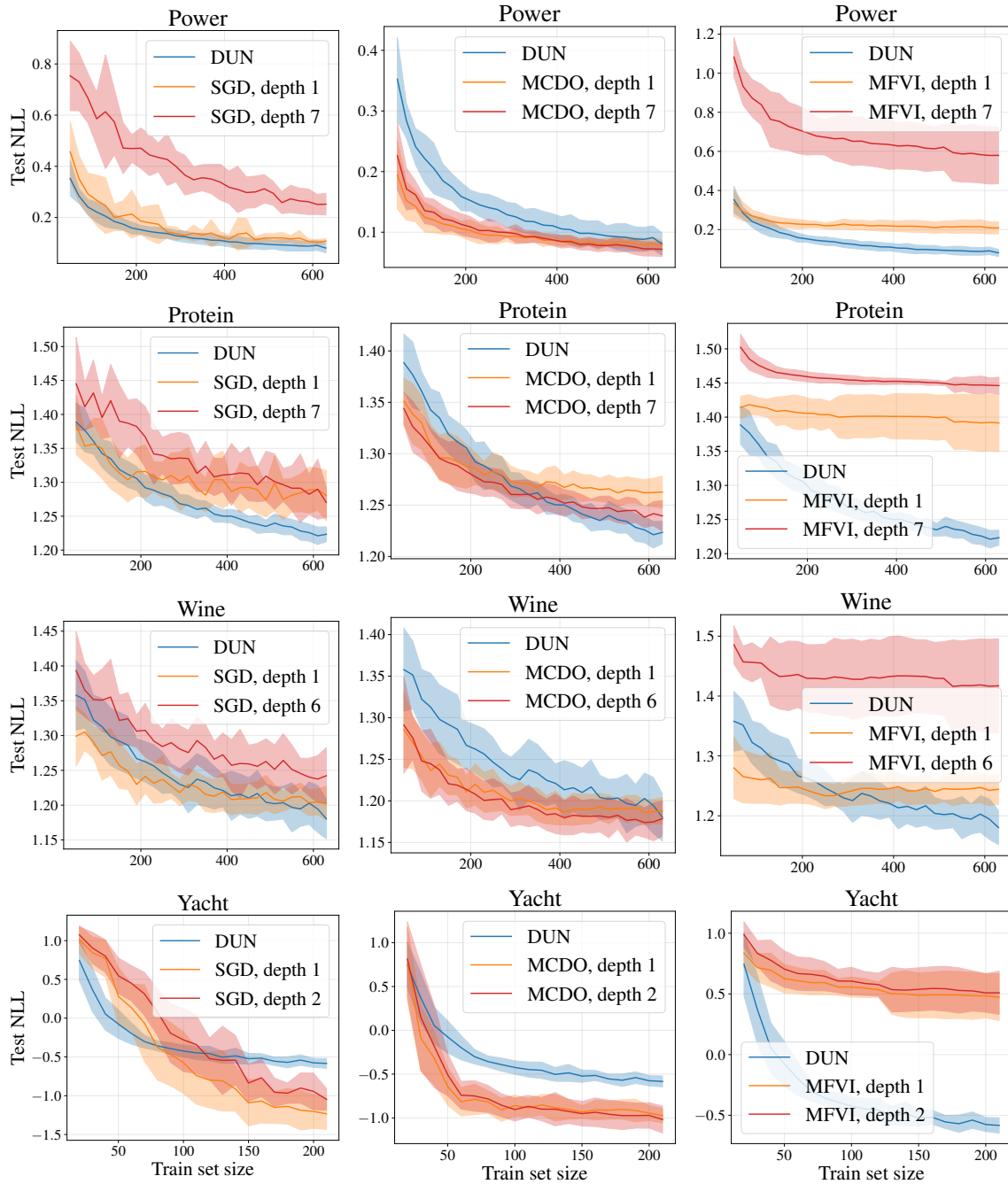
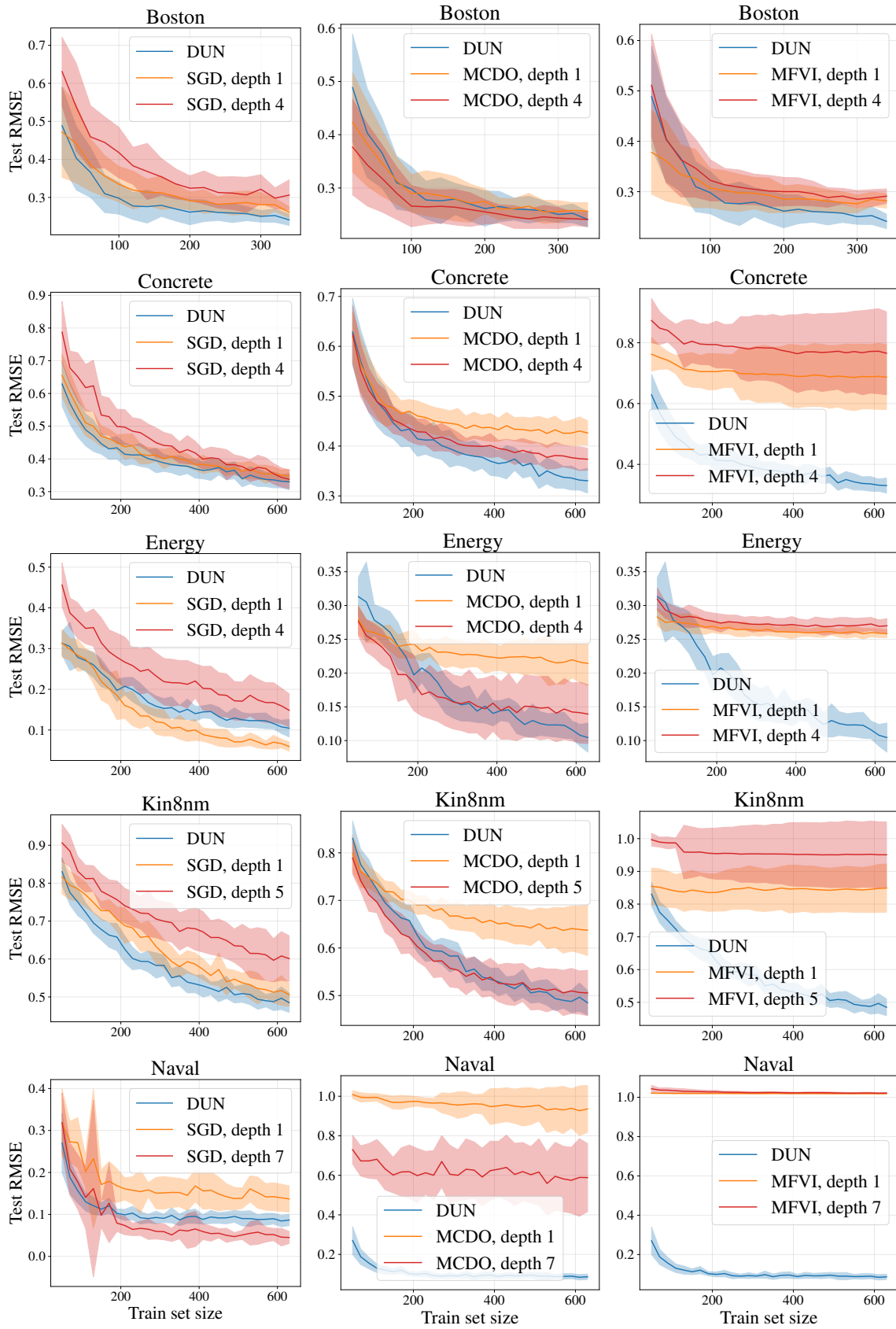


Figure A.10: NLL vs. number of training points evaluated on UCI datasets. The performance of SGD, MCDO and MFVI models with a single hidden layer and with the optimal number of layers found by the DUN is compared to the performance of DUNs.



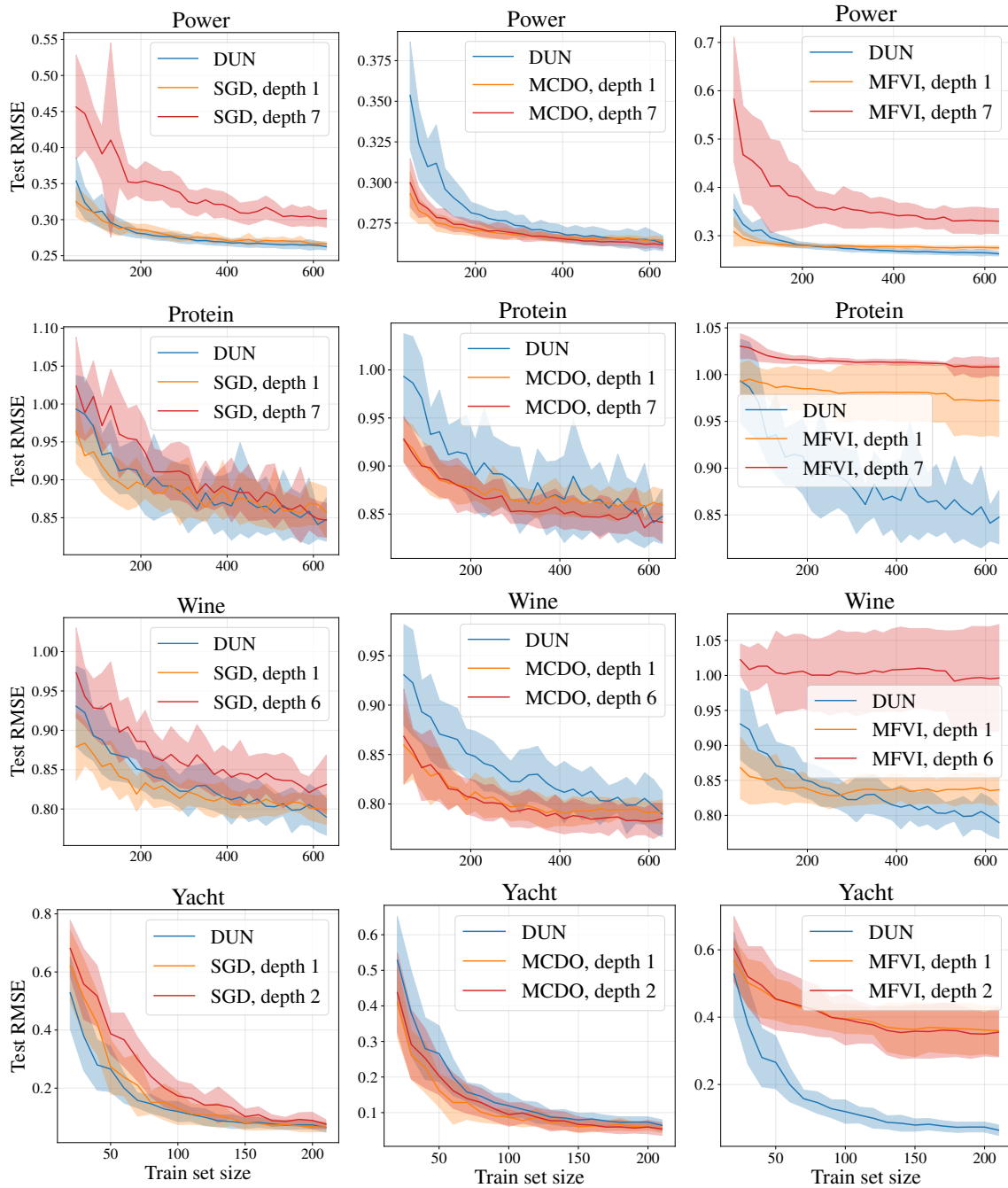


Figure A.11: RMSE vs. number of training points evaluated on UCI datasets. The performance of SGD, MCDO and MFVI models with a single hidden layer and with the optimal number of layers found by the DUN is compared to the performance of DUNs.

