## A Depth uncertainty networks

BNNs translate uncertainty in weight-space into predictive uncertainty by marginalising out the posterior over weights. The intractability of this posterior necessitates the use of approximate inference methods. 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; 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].

DUNs are composed of subnetworks of increasing depth, with each subnetwork contributing one prediction to the final model. The computational model for DUNs is shown in figure A.5. 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 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.

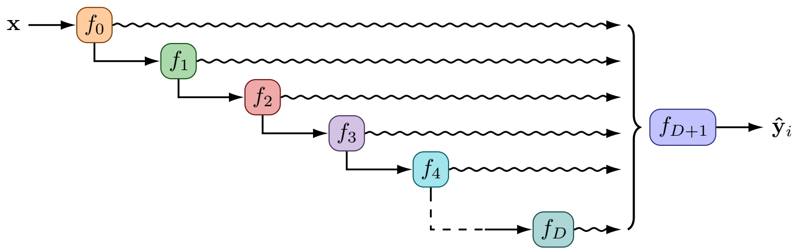

Figure A.5: DUN computational model [Antorán et al., 2020]. Each layer's activations are passed through the output block, producing per-depth predictions.

### A.1 Uncertainty over depth

Figure A.6 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 $\beta$: $p_\beta(d) = \mathrm{Cat}(d \mid \{\beta_i\}_{i=0}^{D})$. The model's marginal log likelihood (MLL) is given by

$$\log p(\mathcal{D}_{\text{train}}; \theta) = \log \sum_{i=0}^{D} \left( p_\beta(d=i) \prod_{n=1}^{N} p\left(\mathbf{y}_n \mid \mathbf{x}_n, d=i; \theta\right) \right), \tag{5}$$

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

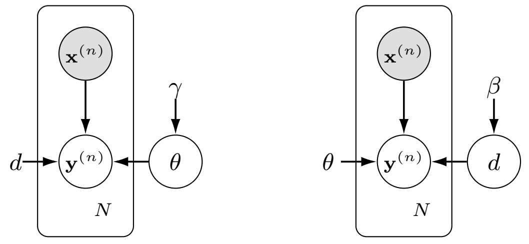

Figure A.6: Left: graphical model of a BNN. Right: graphical model of a DUN.

The posterior,

$$p(d \mid \mathcal{D}_{\text{train}}; \theta) = \frac{p(\mathcal{D}_{\text{train}} \mid d; \theta) p_\beta(d)}{p(\mathcal{D}_{\text{train}}; \theta)}, \tag{6}$$

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 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.

### A.2 Inference in DUNs

Antorán et al. [2020] find that directly optimising the MLL equation (5) 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 variational inference (VI) approach 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 evidence lower bound (ELBO) can be derived:

$$\mathcal{L}(\phi, \theta) = \sum_{n=1}^{N} \mathbb{E}_{q_\phi(d)} \left[\log p\left(\mathbf{y}_n \mid \mathbf{x}_n, d; \theta\right)\right] - \mathrm{KL}\left(q_\phi(d) \| p_\beta(d)\right), \tag{7}$$

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 (7) can be computed exactly, and that optimising $\mathcal{L}(\phi, \theta)$ is equivalent to exact inference of the true posterior $p(d \mid \mathcal{D}_{\text{train}}; \theta)$ in the limit: since both $q$ and $p$ are categorical, equation (7) is convex and tight at the optimum (i.e., $q_\phi(d) = p(d \mid \mathcal{D}_{\text{train}}; \theta)$). Since the expectation term can be computed exactly using the activations at each layer (recall $p(\mathbf{y} \mid \mathbf{x}, d = i; \theta) = p\left(\mathbf{y} \mid f_{D+1}\left(\mathbf{a}_i; \theta\right)\right)$), the ELBO can be evaluated exactly without Monte Carlo sampling.

## B Experimental setup

### B.1 Data

We implement batch-based active learning, with batches of 20 data points acquired in each query (with the exception of Yacht, which has a query size of 10 due to its smaller size). An initial training set representing a small proportion of the full data is selected uniformly at random in the first query. For all datasets, 80% of the data are used for training, 10% for validation and 10% for testing. Details about dataset sizes, input dimensionality, and active learning specifications for each dataset are provided in table B.1.

Table B.1: Summary of datasets and active learning specifications. 80% of the data are used for training, 10% for validation and 10% for testing.

| NAME | SIZE | INPUT DIM. | INIT. TRAIN SIZE | NO. QUERIES | QUERY SIZE |
|---|---|---|---|---|---|
| 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 |

### B.2 Model specifications

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. 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 evidence lower bound on the validation set is used for reporting final results. For MCDO models a fixed dropout probability of $0.1$ is used and prediction is based on 10 MC samples. DUNs use a uniform categorical prior over depth. The hyperparameter settings largely follow those used in Antorán et al. [2020] for their experiments with toy regression datasets.

All experiments are repeated 40 times with different weight initialisations and train-test splits (with the exception of the Protein dataset, for which experiments are repeated only 30 times due to the cost of evaluation on the larger test set). Unless otherwise specified, we report the mean and standard deviation of the relevant metric over the repeated experiment runs. Following Farquhar et al. [2021], the proposal distribution $\alpha\left(i_m; i_{1:m-1}, \mathcal{D}_{\text{pool}}\right)$ used is a stochastic relaxation of the Bayesian active learning by disagreement (BALD) acquisition function proposed by [Houlsby et al., 2011]. BALD scores are multiplied by a constant temperature $T$ before being passed through the `softmax` function; candidate points are then sampled with probability proportional to the `softmax` probabilities. We use $T = 10$ as this is found to yield the best performance for most of the datasets considered (results for experiments to determine the optimal value of $T$ are given in appendix C.4).

## C Additional experimental results

### C.1 Quantifying active learning bias

Figure C.1 presents the results of the active learning bias experiments, as in figure 2, for the remaining datasets. Figure C.2 shows equivalent results for BNNs trained with MCDO.

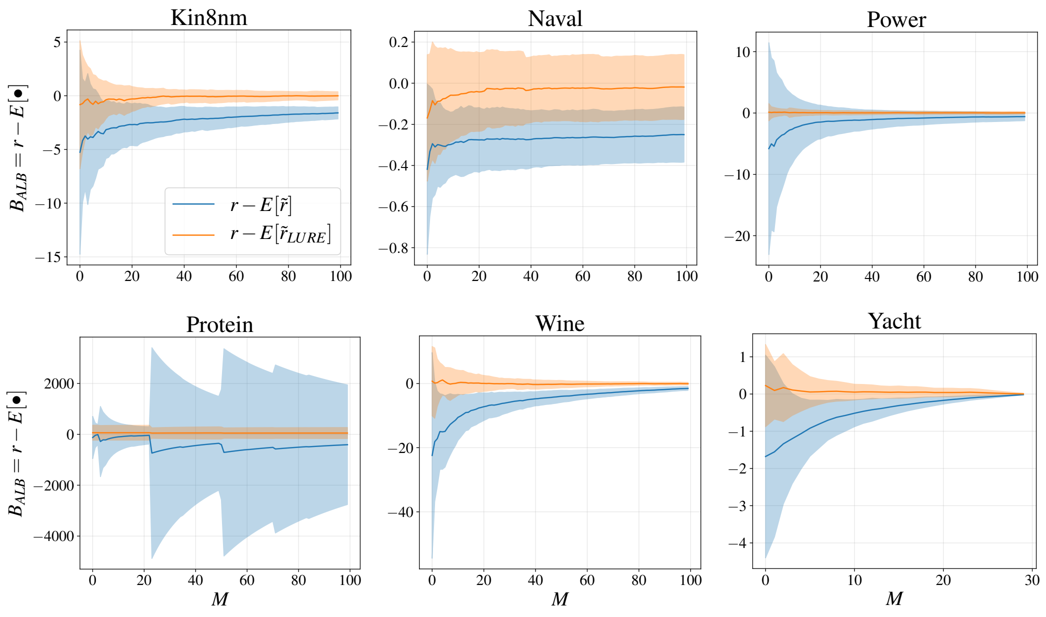

Figure C.1: Bias introduced by actively sampling points with DUNs, evaluated using $\tilde{R}$ (blue) and $\tilde{R}_{\text{LURE}}$ (orange).

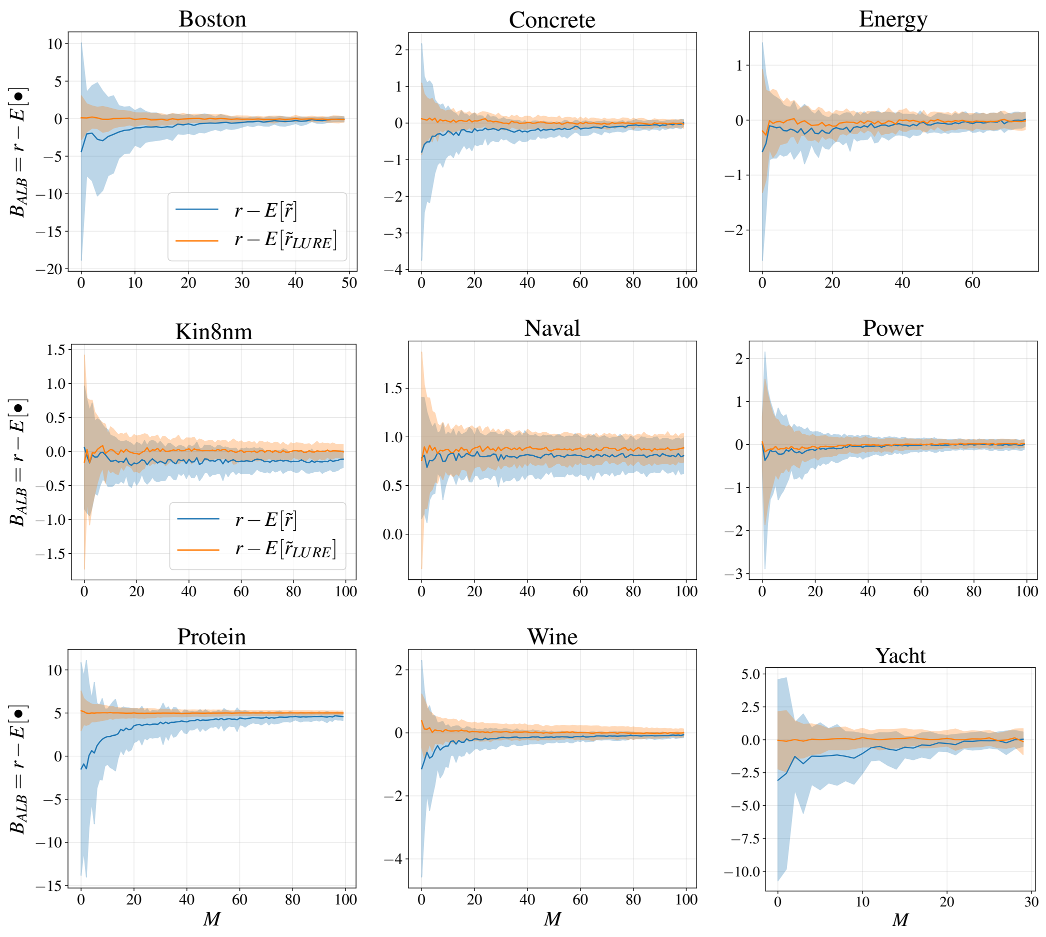

Figure C.2: Bias introduced by actively sampling points with MCDO, evaluated using $\tilde{R}$ (blue) and $\tilde{R}_{\text{LURE}}$ (orange).

## C.2 Overfitting bias

Figure C.3 compares the magnitude of overfitting bias for DUNs and BNNs trained with MCDO, as in figure 3. Figure C.3 shows additionally the overfitting bias when training with $\tilde{R}$.

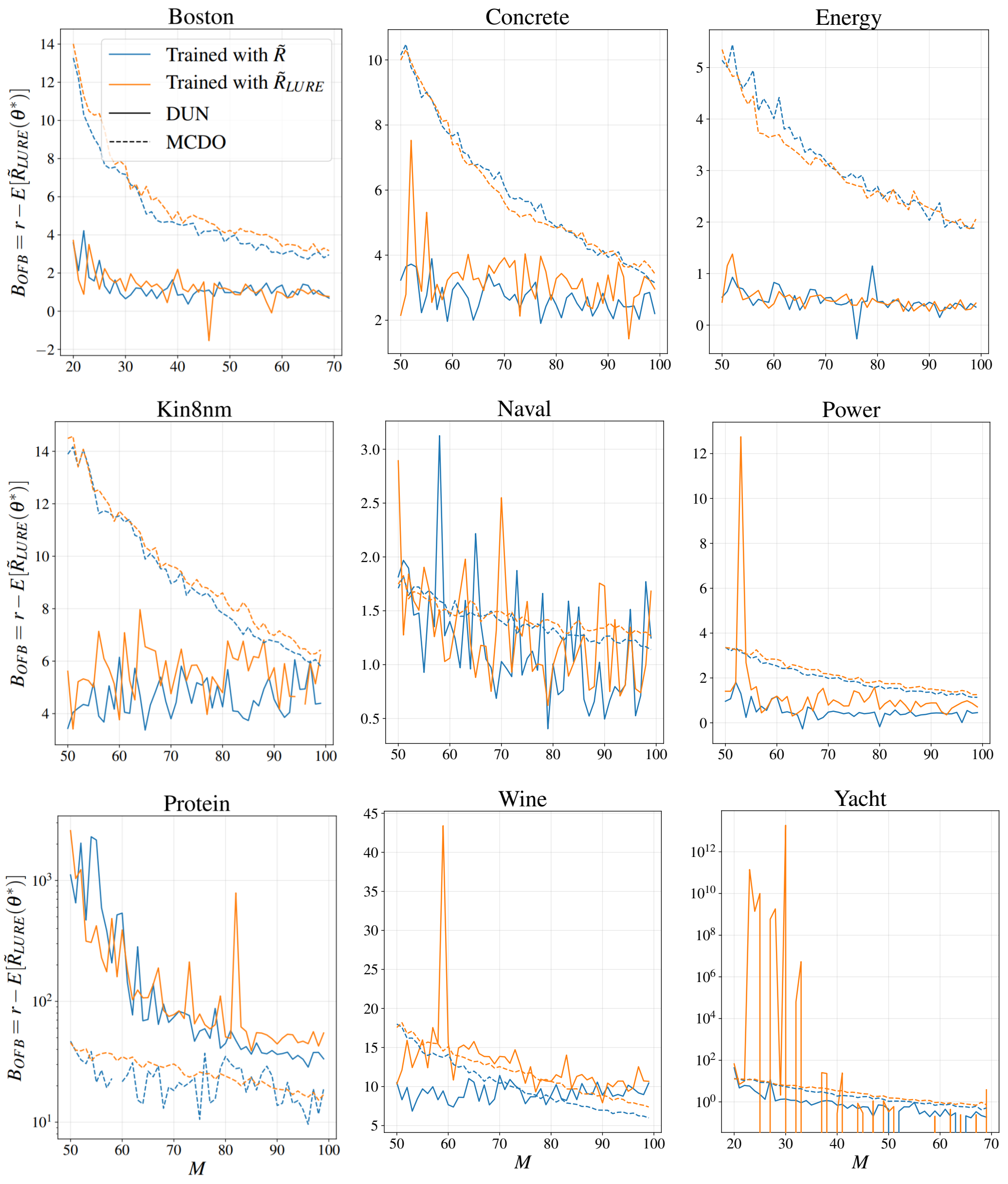

Figure C.3: Overfitting bias for DUNs (solid lines) and MCDO (dashed lines) trained with $\tilde{R}$ (blue) and $\tilde{R}_{\mathrm{LURE}}$ (orange). NLL for Protein and Yacht is displayed in log scale.

## C.3 Training with unbiased risk estimators

Figure C.4 presents the results of training DUNs with $\tilde{R}$ and $\tilde{R}_{\text{LURE}}$, as in figure 4, for the remaining datasets.

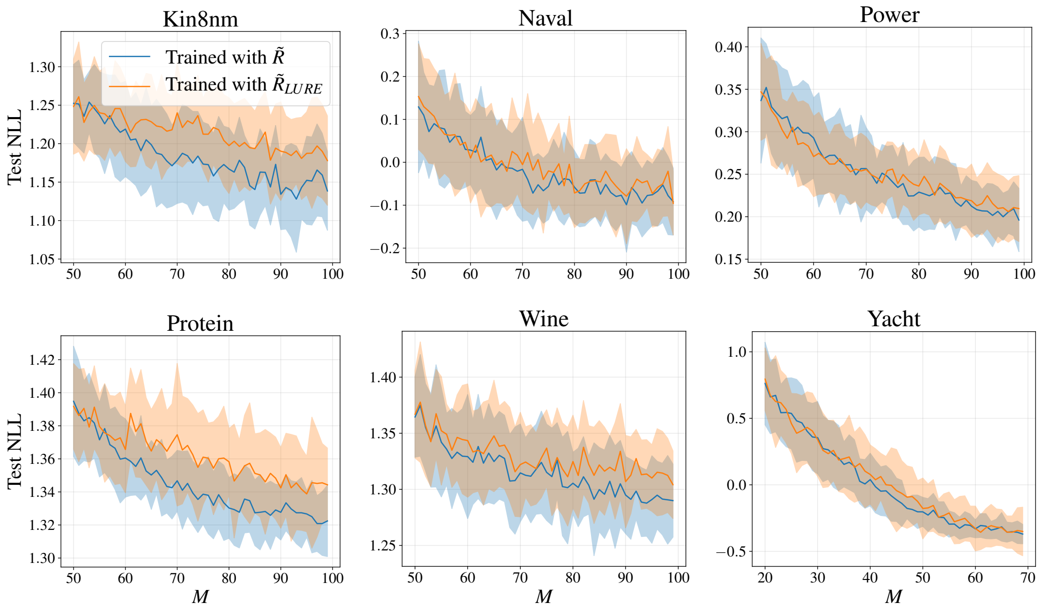

Figure C.4: Test NLL for DUNs trained with $\tilde{R}$ (blue) and $\tilde{R}_{\text{LURE}}$ (orange). A larger value of the orange line indicates that training with $\tilde{R}_{\text{LURE}}$ harms performance, despite removing active learning bias.

## C.4 Temperature of the proposal distribution

Figure C.5 shows the test NLL for DUNs using different temperatures $T$ for the proposal distribution.

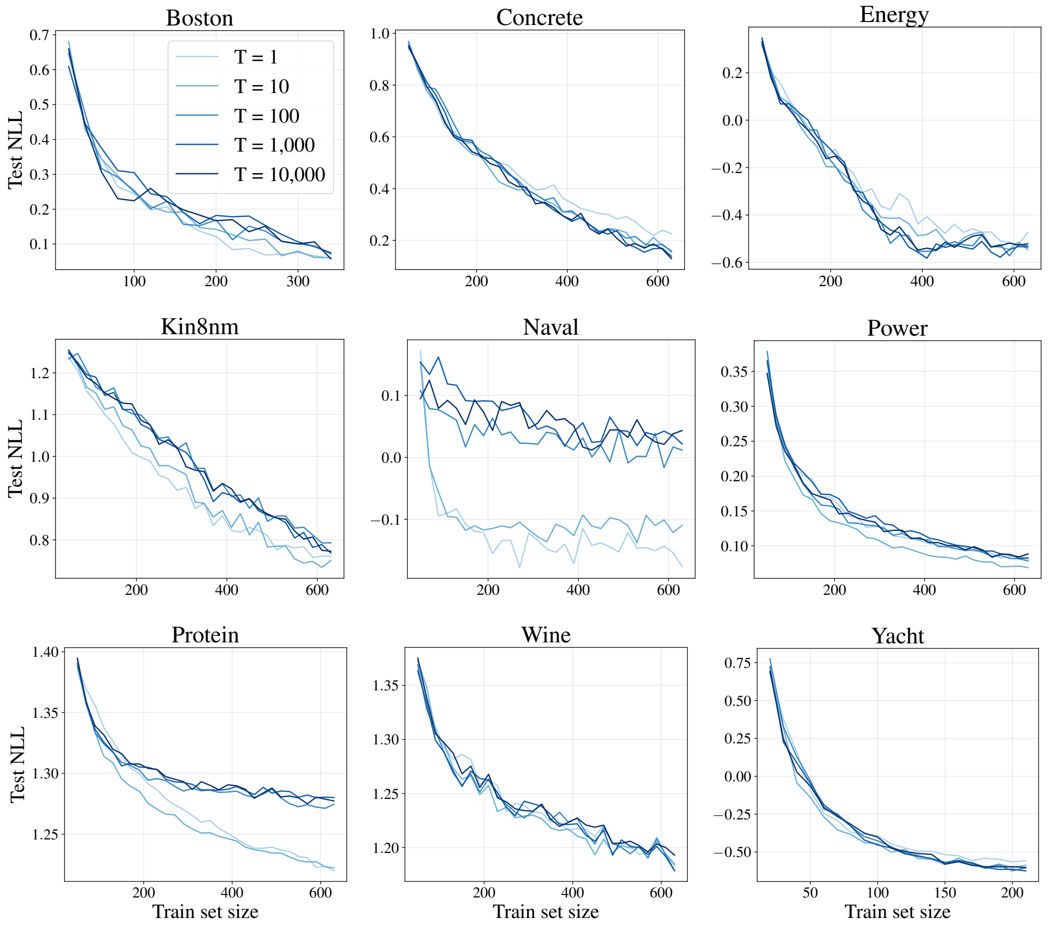

Figure C.5: Test NLL of DUNs using a stochastic relaxation of BALD. Different temperatures of the proposal distribution are compared. Means of 40 runs of the experiments are shown; standard deviations are not plotted for clarity.

## C.5 Depth posteriors for DUNs at different stages of active learning

Figure C.6 shows the posterior distributions over depth for DUNs for the smallest and largest dataset sizes used during active learning.

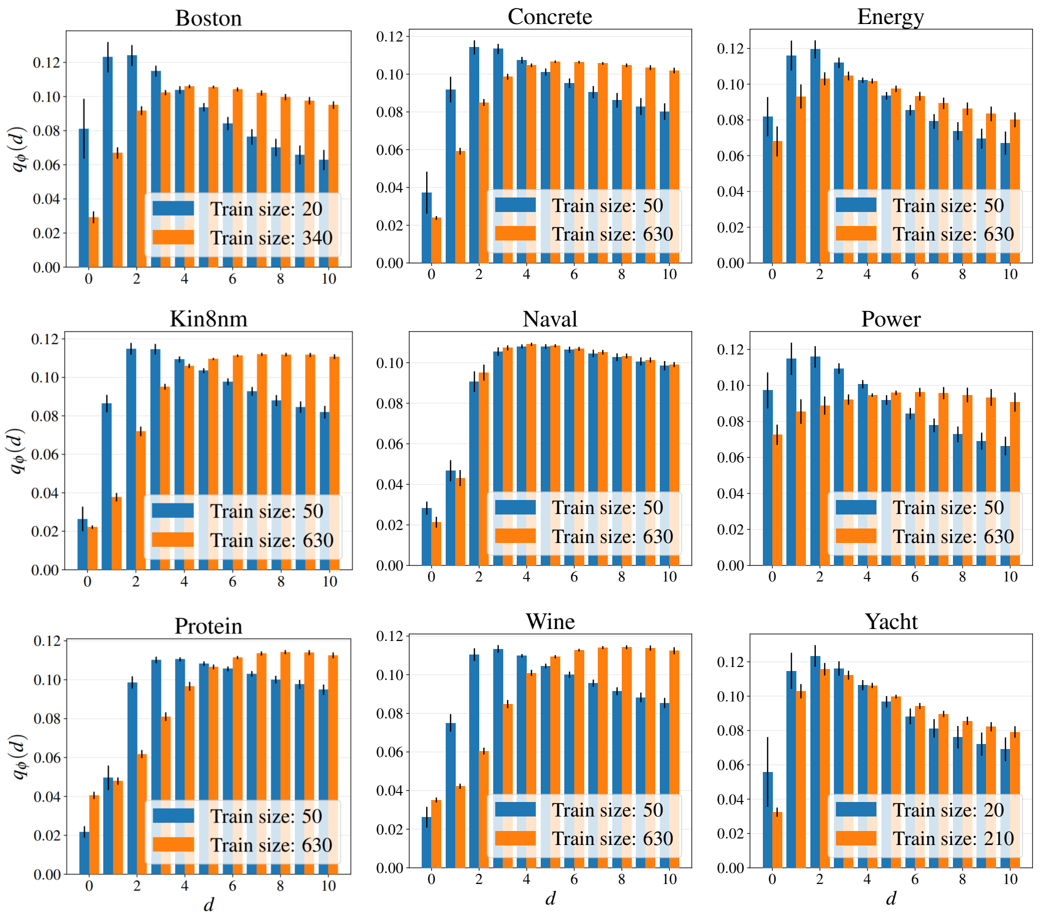

Figure C.6: Posterior probabilities over depth for DUNs trained on UCI regression datasets, for the smallest (blue bars) and largest (orange bars) labelled datasets used in active learning.