AutoDEUQ: Automated Deep Ensemble with Uncertainty Quantification

Romain Egele, Romit Maulik, Krishnan Raghavan, Bethany Lusch, Isabelle Guyon, Prasanna Balaprakash

AutoDEUQ: Automated Deep Ensemble with Uncertainty Quantification

Romain Egele[1, 2], **Romit Maulik**[1], **Krishnan Raghavan**[1], **Bethany Lusch**[1], **Isabelle Guyon**[2]
Prasanna Balaprakash[1]

[1] Argonne National Laboratory
[2] Université Paris-Saclay (CNRS, INRIA, LISN)
romainegele@gmail.com, rmaulik@anl.gov, kraghavan@anl.gov, blusch@anl.gov
isabelle.guyon@universite-paris-saclay.fr, pbalapra@anl.gov

Abstract

Deep neural networks are powerful predictors for a variety of tasks. However, they do not capture uncertainty directly. Using neural network ensembles to quantify uncertainty is competitive with approaches based on Bayesian neural networks while benefiting from better computational scalability. However, building ensembles of neural networks is a challenging task because, in addition to choosing the right neural architecture or hyperparameters for each member of the ensemble, there is an added cost of training each model. We propose AutoDEUQ, an automated approach for generating an ensemble of deep neural networks. Our approach leverages joint neural architecture and hyperparameter search to generate ensembles. We use the law of total variance to decompose the predictive variance of deep ensembles into aleatoric (data) and epistemic (model) uncertainties. We show that AutoDEUQ outperforms probabilistic backpropagation, Monte Carlo dropout, deep ensemble, distribution-free ensembles, and hyper ensemble methods on a number of regression benchmarks.

Introduction

Uncertainty quantification (UQ) for machine-learning-based predictive models is crucial for assessing the trustworthiness of predictions from the trained model. For deep neural networks (NNs), it is desirable for predictions to be accompanied with estimates of uncertainty due to the black-box nature of the function approximation. There are two major forms of uncertainty (Hüllermeier and Waegeman 2021): aleatoric data uncertainty and epistemic model uncertainty. The former occurs due to the inherent variability or noise in the data. The latter is attributed to the uncertainty associated with the NN model parameter estimation or out-of-distribution predictions. The epistemic uncertainty increases in the regions that are not well represented in the training dataset (Gal and Ghahramani 2016a). While the aleatoric uncertainty is irreducible, the epistemic uncertainty can be reduced by collecting more training data in the appropriate regions.

There are several instances of research that have looked at extending deterministic neural networks to probabilistic models. A strongly advocated method is to have a fully Bayesian formulation, where each trainable parameter in a DNN is assumed to be sampled from a very high-dimensional (and arbitrary) joint distribution (Neal 2012). However, this is computationally infeasible, for example due to issues of convergence, for any practical deep learning tasks with millions of trainable parameters in the architecture and having large datasets. Consequently, several approximations to fully Bayesian formulations have been put forth to reduce the computational complexity of uncertainty quantification in DNNs. These range from simple augmentations such as the mean-field approximation in Bayesian backpropagation via variational inference (Hoffman et al. 2013; Hernández-Lobato and Adams 2015), where each parameter is assumed to be sampled from an independent unimodal Gaussian distribution, to Monte-Carlo dropout (Srivastava et al. 2014) where random neurons are switched off during training and inference to obtain ensemble predictions.

In recent studies, ensemble methods that utilize multiple independently trained DNNs have shown considerable promise for providing effective uncertainty quantification (Lakshminarayanan, Pritzel, and Blundell 2017; Ovadia et al. 2019; Ashukha et al. 2020) by outperforming conventional approximations to the fully Bayesian methodology. Wilson and Izmailov (2020) argue that the deep ensembles approach is fully congruous with Bayesian model averaging, which attempts to estimate the posterior distribution of the targets given input data by marginalizing the parameters. However, a key factor in deep ensembles is model diversity without which uncertainty cannot be captured efficiently. For example, in (Lakshminarayanan, Pritzel, and Blundell 2017), each member of the ensemble has an identical neural architecture and is trained using maximum likelihood or maximum a posteriori optimization through different initialization of weights. Consequently, ensemble diversity is limited as each model can at best settle on distinct local minimas. Marginalization over these models in the ensemble will force the function approximation to collapse on one hypothesis and provide results similar to Bayesian model averaging for a single architecture with probabilistic trainable parameters. Such an implicit assumption may be undesirable when dealing with datasets that are generated from a combination of hypotheses. Moreover, the lack of flexibility in the ensemble may lead to a poorer estimate of epistemic uncertainty. Although Wenzel et al. (2021) attempt to relax this issue by allowing more diversity in the ensemble, they vary just two hyperparameters. Similarly, Zaidi et al. (2020) varied the architecture with fixed

trainable hyperparameters to increase the ensemble diversity. By constructing diverse DNNs models through a methodical and automated approach, we hypothesize that the assumption of, and the eventual collapse, to one hypothesis can be avoided, thus providing robust and efficient estimates of uncertainty. We accomplish this by relaxing the requirement for the same neural architecture for each ensemble member or for having fixed trainable hyperparameters for all the ensemble members.

Related Work

To model aleatoric uncertainty, one must model the conditional distribution $p(y \mid \mathbf{x})$ for the target y given an input $\mathbf{x}$. One way is to assume that this distribution is Gaussian and then estimate its parameters (mean and variance) (Nix and Weigend 1994). However, these estimates summarise conditional distributions into scalar values, and are thus unable to model more complex profiles of uncertainty such as multimodal or heteroscedastic. To resolve this issue, implicit generative models (Mohamed and Lakshminarayanan 2016) and mixture density networks (Bishop 1994) can be employed. A different approach is deep kernel learning (van Amersfoort et al. 2021) which extracts kernels and uses them in Gaussian-process-based methods for datasets with large features and sample size. However, this adds additional complexity because one must find the correct hyperparameters. An alternative strategy is to directly output prediction intervals from the NN, such as in (Pearce et al. 2018) which has the advantage of not requiring any distribution assumption on the output variables. However, these methods are ill-equipped to quantify epistemic uncertainty.

Several methods for epistemic uncertainty have been proposed. Bayesian NNs (BNN) (Maddox et al. 2019) and deep ensembles (Caruana, Munson, and Niculescu-Mizil 2006) are the main approaches. In BNN, the weights are assumed to follow a joint distribution and the epistemic uncertainty is quantified through Bayesian inference. However, except for the trivial cases, Bayesian inference is computationally intractable. Therefore, several approximations to BNN have been proposed, such as Probabilistic Backpropagation (PBP) (Hernández-Lobato and Adams 2015) and Bayes by Backprop (Blundell et al. 2015). In deep ensembles (Lakshminarayanan, Pritzel, and Blundell 2017), multiple networks are aggregated to quantify the uncertainty. Each network in the ensemble provides an estimate of aleatoric uncertainty while their aggregation provides an estimate of epistemic uncertainty. However, the members of such ensembles often have similar architecture and hyperparameter values but with different weights generated through random weight initialization in addition to the stochastic aspect of the training procedure. Recently, new automated methods were proposed to improve deep ensembles, wherein hyperparameters (Wenzel et al. 2021) or neural architecture decision variables (Zaidi et al. 2020) are varied to improve the diversity of models in the ensemble to achieve improved aleatoric and epistemic uncertainty estimates.

Recently, Russell and Reale (2021) developed a joint covariance matrix with end-to-end training using a Kalman filter to represent aleatoric uncertainty while using dropout to estimate the epistemic component. While not an ensemble method, it models aleatoric and epistemic at the same time.

Contributions

Given training and validation data, the proposed AutoDEUQ method i) starts from a user-defined neural architecture and hyperparameter search space; ii) leverages aging evolution and Bayesian optimization methods to automatically tune the architecture decision variables and training hyperparameters, respectively; iii) builds a catalog of models from the search; and iv) uses a greedy heuristic to select models from the catalog to construct ensembles. The predictions from the ensemble models are then used to estimate the aleatoric and epistemic uncertainty. AutoDEUQ is built on the successes of three recent works in the deep ensemble literature: deep ensemble (Lakshminarayanan, Pritzel, and Blundell 2017), hyper ensemble (Wenzel et al. 2021), and neural ensemble search (Zaidi et al. 2020). However, our AutoDEUQ method differs from deep ensemble in the following ways: while aleatoric and epistemic uncertainties are modeled empirically, we theoretically decompose the predicted variance of deep ensembles into its aleatoric and epistemic components. Moreover, in AutoDEUQ, the DNN architectures and the training hyperparameter values in the ensembles are different and more importantly they are generated automatically. While hyper ensemble and neural ensemble methods explore hyperparameters and architectural choices, respectively, and generate ensembles, AutoDEUQ explores both spaces simultaneously. In a nutshell, the key contributions of the paper are:

1. variance decomposition to separate aleatoric and epistemic uncertainty estimates from the predictive variance of deep ensembles;
2. automation of deep ensembles construction with joint neural architecture and hyperparameter search; and
3. demonstration of improved uncertainty quantification compared to prior ensemble methods and consequently, advancement of state of the art in deep ensembles.

AutoDEUQ

We focus on uncertainty estimation in a regression setting. Our methodology, automated deep ensemble for uncertainty quantification (AutoDEUQ) estimates aleatoric and epistemic uncertainties by:

1. automatically generating a catalog of NN models through joint neural architecture and hyperparameter search, wherein each model is trained to minimize the negative log likelihood to capture aleatoric uncertainty; and
2. selecting a set of models from the catalog to construct the ensembles and model epistemic uncertainty without losing the quality of aleatoric uncertainty.

Variance decomposition for deep ensembles

In supervised learning, the dataset $\mathcal{D}$ is composed of i.i.d points $(\mathbf{x}_i \in \mathcal{X}, y_i \in \mathcal{Y})$, where $\mathbf{x}_i$ and $y_i = f(\mathbf{x}_i)$ are the input and the corresponding output of the i-th point, respectively, and $\mathcal{X} \subset \mathbb{R}^N$ and $\mathcal{Y} \subset \mathbb{R}^M$ are the input and output

spaces of N and M dimensions, respectively. Here, we focus on regression problems, wherein the output is a scalar or vector of real values. Given $\mathcal{D}$, we seek to model the probabilistic predictive distribution $p(y|\mathbf{x})$ using a parameterized distribution $p_\theta(y|\mathbf{x})$, which estimates aleatoric uncertainty through a trained NN and then estimates the epistemic uncertainty with an ensemble of NNs $p_\mathcal{E}(y|\mathbf{x})$. We define Θ to be the sample space for θ.

The aleatoric uncertainty can be modeled using the quantiles of p_θ. Following previous works (Lakshminarayanan, Pritzel, and Blundell 2017), we assume a Gaussian distribution for $p_\theta \sim \mathcal{N}(\mu_\theta, \sigma_\theta^2)$ and use variance as a measure of the aleatoric uncertainty. We explicitly partition θ into $(\theta_a, \theta_h, \theta_w)$ such that Θ is decomposed into $(\Theta_a, \Theta_h, \Theta_w)$, where $\theta_a \in \Theta_a$ represents the NN values of the architecture decision variables (network topology parameters), $\theta_h \in \Theta_h$ represents NN training hyperparameters (e.g. learning rate, batch size), and $\theta_w \in \Theta_w$ represents the NN weights. The NN is trained to output mean μ_θ and variance σ_θ^2. For a given choice of architecture decision variables θ_a and training hyperparameters θ_h, to obtain θ_w^*, we seek to maximise the likelihood given the real data $\mathcal{D}$. Specifically, we can model the aleatoric uncertainty using the negative log-likelihood loss (as opposed to the usual mean squared error) in the training (Lakshminarayanan, Pritzel, and Blundell 2017):

$$\ell(\mathbf{x}, y; \theta) = -\log p_\theta = \frac{\log \sigma_\theta^2(\mathbf{x})}{2} + \frac{(y - \mu_\theta(\mathbf{x}))^2}{2\sigma_\theta^2(\mathbf{x})} + \text{cst}, \tag{1}$$

where cst is a constant. The NN training problem is then

$$\theta_w^* = \underset{\theta_w \in \Theta_w}{\arg\max} \quad \ell(\mathbf{x}, y; \theta_a, \theta_h, \theta_w). \tag{2}$$

To model epistemic uncertainty, we use deep ensembles (an ensemble composed of NNs) (Lakshminarayanan, Pritzel, and Blundell 2017). In our approach, we generate a catalog of NN models $\mathcal{C} = \{\theta_i, i = 1, 2, \cdots, c\}$ (where $\theta \in \Theta$ is a tuple of architecture, optimization hyperparameters and weights) and select K models to form the ensemble. Let $p(.)$ be a probability measure such that $p : \Theta \to [0, 1] \subset \mathbb{R}$ and $p(\theta)$ describes the probability of $\theta \in \mathcal{E}$, where $\mathcal{E}$ is the set of θ present in the ensemble such that $\mathcal{E} = \{\theta_i, i = 1, 2, \cdots, K\}$ with K being the size of the ensemble. The overall probability density function of the ensemble is obtained using the mixture distribution (mixture of all the members in the ensemble) $p_\mathcal{E}$ given as

$$p_\mathcal{E} = \mathbb{E}_{\theta \sim p(\mathcal{E})} p_\theta, \tag{3}$$

where $p(\mathcal{E})$ refers to a probability distribution over the ensemble.

Proposition 1. *Define $\mathcal{E}$ be the ensemble and Θ be the sample space for θ. Let $p : \Theta \to [0, 1] \subset \mathbb{R}$ and define $p(\mathcal{E})$ refers to a probability distribution over the ensemble. Define μ_θ and σ_θ^2 as the mean and variance of p_θ for each $\theta \in \mathcal{E}$. Define the probability density function $p_\mathcal{E}$ as in Eq.* (3)*. Then the mean of $p_\mathcal{E}$ is*

$$\mu_\mathcal{E} := \mathbb{E}_{\theta \sim p(\mathcal{E})}[\mu_\theta], \tag{4}$$

and the variance is

$$\sigma_\mathcal{E}^2 := \mathbb{V}_{\theta \sim p(\mathcal{E})}[p_\mathcal{E}] = \underbrace{\mathbb{E}_{\theta \sim p(\mathcal{E})}[\sigma_\theta^2]}_{\textit{Aleatoric Uncertainty}} + \underbrace{\mathbb{V}_{\theta \sim p(\mathcal{E})}[\mu_\theta]}_{\textit{Epistemic Uncertainty}}, \tag{5}$$

where $\mathbb{E}$ refers to the expected value and $\mathbb{V}$ refers to the variance.

Proof. By the definition of the mean for mixture distributions (McLachlan, Lee, and Rathnayake 2019)

$$\mu_\mathcal{E} := \mathbb{E}_{\theta \sim p(\mathcal{E})}[\mu_\theta] \tag{6}$$

To obtain variance of the mixture distribution, we will write

$$\begin{aligned} \sigma_\mathcal{E}^2 &:= \mathbb{E}_{\theta \sim p(\mathcal{E})}[(y - \mu_\mathcal{E})^2] \\ &= \mathbb{E}_{\theta \sim p(\mathcal{E})}[y^2] - \mu_\mathcal{E}^2 \\ &= (\sum_{\theta \in \mathcal{E}} \mathbb{E}[y^2]) - \mu_\mathcal{E}^2 \\ &= (\sum_{\theta \in \mathcal{E}} (\sigma_\theta^2 + \mu_\theta^2)) - \mu_\mathcal{E}^2 \end{aligned} \tag{7}$$

Finally, by definition of mean and variance and applying the law of total variance provides our result as

$$\sigma_\mathcal{E}^2 = \mathbb{E}_{\theta \sim p(\mathcal{E})}[\sigma_\theta^2] + \mathbb{V}_{\theta \sim p(\mathcal{E})}[\mu_\theta] \tag{8}$$

□

Eq. (5) formally provides the decomposition of overall uncertainty of the ensemble into its individual components such that

- $\mathbb{E}_{\theta \sim p(\mathcal{E})}[\sigma_\theta^2]$ marginalizes the effect of θ and captures the aleatoric uncertainty.
- $\mathbb{V}_{\theta \sim p(\mathcal{E})}[\mu_\theta]$ captures the spread of the prediction across different models and neglects the noise of the data, therefore capturing the epistemic uncertainty.

Assuming that $p(\mathcal{E})$ is uniform in the mixture distribution (i.e., equal weights), we can use the empirical mean and variance estimates to compute the two quantities in Eq. (5) such that

$$\begin{aligned} \mu_\mathcal{E} &= \frac{1}{K} \sum_{\theta \in \mathcal{E}} \mu_\theta \\ \sigma_\mathcal{E}^2 &= \underbrace{\frac{1}{K} \sum_{\theta \in \mathcal{E}} \sigma_\theta^2}_{\text{Aleatoric Uncertainty}} + \underbrace{\frac{1}{K-1} \sum_{\theta \in \mathcal{E}} (\mu_\theta - \mu_\mathcal{E})^2}_{\text{Epistemic Uncertainty}} \end{aligned} \tag{9}$$

where K is the size of the ensemble.

Here, we derived that the total uncertainty quantified by $\sigma_\mathcal{E}^2$ is a combination of aleatoric and epistemic uncertainty, which are given by the the mean of the predictive variance of each model in the ensemble and the predictive variance of the mean of each model in the ensemble.

Catalogue generation and ensemble construction

Let $\mathcal{D}$ be decomposed as $\mathcal{D} = \mathcal{D}^{train} \cup \mathcal{D}^{valid} \cup \mathcal{D}^{test}$, referring to the training, validation, and test data, respectively. A neural architecture configuration θ_a is a vector from the neural architecture search space Θ_a, defined by a set of neural architecture decision variables. A hyperparameter configuration θ_h is a vector from the training hyperparameter search space Θ_h defined by a set of hyperparameters used for training (e.g., learning rate, batch size). The problem of joint neural architecture and hyperparameter search can be formulated as the following bi-level optimization problem:

$$\theta_a^*, \theta_h^* = \arg\max_{\theta_a, \theta_h} \frac{1}{N^{valid}} \sum_{\mathbf{x}, y \in \mathcal{D}^{valid}} \ell(\mathbf{x}, y; \theta_a, \theta_h, \theta_w^*)$$
$$\text{s.t. } \theta_w^* = \arg\max_{\theta_w} \frac{1}{N^{train}} \sum_{\mathbf{x}, y \in \mathcal{D}^{train}} \ell(\mathbf{x}, y; \theta_a, \theta_h, \theta_w), \tag{10}$$

where the best architecture decision variables θ_a^* and training hyperparameters values θ_h^* are selected based on the validation set and the corresponding weights θ_w are selected based on the training set.

The pseudo code of the AutoDEUQ is shown in Algorithm 1. To perform a joint neural architecture and hyperparameter search, we leverage aging evolution with Bayesian optimization (AgEBO) (Egele et al. 2021). This method combines aging evolution (AgE) (Real et al. 2018), a parallel neural architecture search (NAS) method for searching over the architecture space, and asynchronous Bayesian optimization (BO) for tuning the training hyperparameters. The AgEBO method follows the manager-worker paradigm, wherein a manager node runs a search method to generate multiple NNs and W workers (compute nodes) train them simultaneously. The AgEBO method constructs the initial population by sampling W architecture and W hyperparameter configurations and concatenating them (l. 1–7). The NNs obtained by using these concatenated configurations are sent for simultaneous evaluation on W workers (l. 6). The iterative part (l. 8–26) of the method checks if any of the workers finish their evaluation (l. 9), collects validation metric values from the finished workers, and uses them to generate the next set of architecture and hyperparameter configurations for simultaneous evaluation to fill up the free workers that finished their evaluations (l. 11–25). At a given iteration, to generate a NN, architecture and hyperparameter configurations are generated in the following way: from the incumbent population, S NNs are sampled (l. 17). A random mutation is applied to the best of S NNs to generate a child architecture configuration (l. 18). This mutation is obtained by first randomly selecting an architecture decision variable from the selected NN and replacing its value with another randomly selected value excluding the current value. The new child replaces the oldest member of the population. The AgEBO optimizes the hyperparameters (θ_h) by marginalizing the architecture decision variables (θ_a). At a given iteration, to generate a hyperparameter configuration, the AgEBO uses a (supervised learning) model M to predict a point estimate (mean value) $\mu(\theta_h^i)$ and standard deviation $\sigma(\theta_h^i)$ for a large number of unevaluated hyperparameter configurations. The best configuration is selected by ranking all sampled hyperparameter configurations using the upper confidence bound (UCB) acquisition function, which is parameterized by $\kappa \geq 0$ that controls the trade-off between exploration and exploitation. To generate multiple hyperparameter configurations at the

Algorithm 1: AutoDEUQ for ensemble construction

inputs : P: population size, S: sample size, W: workers
output : $\mathcal{E}$: ensemble of models

```
/* Initialization for AgEBO                                   */
1  population ← create_queue(P) // Alloc empty Q of size P
2  BO ← Bayesian_Optimizer()
3  for i ← 1 to W do
4      config.θ_a ← random_sample(Θ_a)
5      config.θ_h ← random_sample(Θ_h)
6      submit_for_training(config) // Nonblocking
7  end
   /* Optimization loop for AgEBO                             */
8  while stopping criterion not met do
       // Query results
9      results ← check_finished_training()
10     C ← C ∪ results // Add to catalogue population
11     if |results| > 0 then
12         population.push(results) // Aging population
           // Generate hyperparameter configs
13         BO.tell(results.θ_h, results.valid_score)
14         next ← BO.ask(|results|) // Generate architecture configs
15         for i ← 1 to |results| do
16             if |population| = P then
17                 parent.config ← select_parent(population,S)
18                 child.config.θ_a ← mutate(parent.θ_a)
19             else
20                 child.config.θ_a ← random_sample(Θ_a)
21             end
22             child.config.θ_h ← next[i].θ_h
23             submit_for_training(child.config) // Nonblocking
24         end
25     end
26 end
   /* Initialization for ensemble construction                */
27 E ← {}
28 min_loss ← +∞
   /* Model selection                                         */
29 while |E.unique()| ≤ K do
30     θ* ← arg min_{θ∈C} ℓ(E ∪ {θ}, X, y)
31     if ℓ(E ∪ {θ*}, X, y) ≤ min_loss then
32         E ← E ∪ {θ*}
33         min_loss ← ℓ(E, X, y)
34     else
35         return E
36     end
37 end
38 return E
```

same time, the AgEBO leverages multipoint acquisition function based on a constant liar strategy (Ginsbourger, Le Riche, and Carraro 2010).

The catalog $\mathcal{C}$ of NN models will be obtained by running AgEBO and storing all the models from the runs. To build the ensemble $\mathcal{E}$ of models from $\mathcal{C}$, we adopt a greedy selection strategy (l. 27–38). This approach, originally proposed in (Caruana et al. 2004), consists of iteratively building the ensemble greedily. At each step, the model from the catalog that most improves the negative log likelihood of the incumbent ensemble is added to the ensemble. The greedy approach can work well when the validation data is representative of the generalisation task (i.e., big enough, diverse enough, with good coverage) (Caruana et al. 2004).

Results

We first describe the search space used in AutoDEUQ. Next, using a one-dimensional dataset, we present an ablation study to analyze the impact of different components of AutoDEUQ on uncertainty estimation. Finally, we compare AutoDEUQ with other methods using a set of 10 regression benchmarks.

Search Space

The architecture search space is modeled using a directed acyclic graph, which starts and ends with input and output nodes, respectively (see Appendix for an illustration). They represent the input and output layer of NN, respectively. Between the two are intermediate nodes defined by a series of variable $\mathcal{N}$ and skip-connection $\mathcal{SC}$ nodes. Both types of nodes correspond to categorical decision variables. The variable nodes model dense layers with a list of different layer configurations. The skip connection node creates a skip-connection between the variable nodes. This second type of node can take two values: disable or create the skip connection. For a given pair of consecutive variable nodes $\mathcal{N}_k, \mathcal{N}_{k+1}$, three skip-connection nodes $\mathcal{SC}_{k-3}^{k+1}, \mathcal{SC}_{k-2}^{k+1}, \mathcal{SC}_{k-1}^{k+1}$ are created. These skip-connection nodes allow for connection to the previous nonconsecutive variable nodes $\mathcal{N}_{k-3}, \mathcal{N}_{k-2}, \mathcal{N}_{k-1}$, respectively. Each dense layer configuration is defined by the number of units and the activation function. We used values in {16, 32, ..., 256} and {elu, gelu, hard sigmoid, linear (i.e., identity), relu, selu, sigmoid, softplus, softsign, swish, tanh}, respectively. These resulted in 177 (16 units $\times$ 11 activation functions, and identity) dense layer types for each variable node. Skip connections can be created from at most 3 previous dense layers. Each skip-connection is created with a linear projection so that feature vectors match in shape, then addition to merge the vectors. The number of variable nodes is set to 3 for the one-dimensional toy dataset and to 5 for the regression benchmarks.

For the hyperparameter search space, we use a learning rate in the continuous range $[10^{-4}, 10^{-1}]$ with a log-uniform prior, batch size in the discrete range $[1, 2, 3, \ldots, b_{max}]$ (where $b_{max} = 32$ for the toy example and $b_{max} = 256$ for the benchmark) with a log-uniform prior, optimizer in {sgd, rmsprop, adagrad, adam, adadelta, adamax, nadam}, a patience for the reduction of the learning rate in the discrete range $[10, 11, \ldots, 20]$ and a patience for early stopping

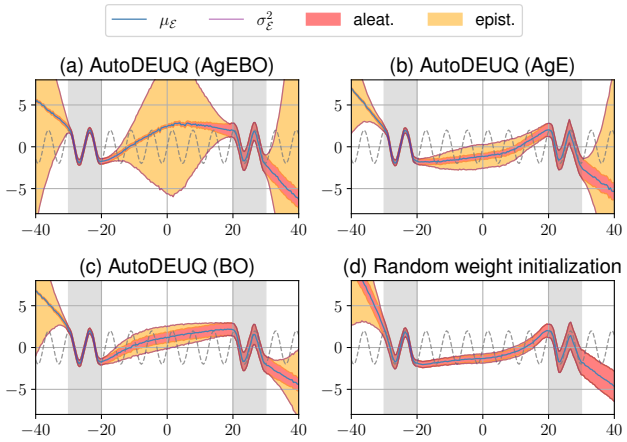

Figure 1: Ablation study of catalog generation: We progressively removed the different algorithmic components of AutoDEUQ and analyzed their impact on the uncertainty estimation.

in the discrete range $[20, 21, \ldots, 30]$. The NNs are trained with 200 epochs for the toy example and 100 epochs for the benchmark. The search space is the same for the toy and the benchmark. Models are check-pointed during their evaluation based on the minimum validation loss achieved. Input and output variables are standardized to have a mean of 0 and a unit variance.

The hardware and software platforms used for the experiments as well as other execution settings are described in the appendix.

Toy Example

We follow the ideas from (Hernández-Lobato and Adams 2015) to assess qualitatively the effectiveness of AutoDEUQ on a one-dimensional dataset. However, instead of the unimodal dataset generated from the cubic function used in (Hernández-Lobato and Adams 2015), we used the $y = f(x) = 2\sin x + \epsilon$ sine function. We generated 200 points randomly sampled from a uniform prior in the x-range $[-30, -20]$ with $\epsilon \sim \mathcal{N}(0, 0.25)$ and 200 other points randomly sampled in the x-range $[20, 30]$ with $\epsilon \sim \mathcal{N}(0, 1)$. These 400 points constitute $\mathcal{D}^{train} \cup \mathcal{D}^{valid}$. We used random sampling to split the generated data: 2/3 for training and 1/3 for validation datasets. The two x-ranges are sampled with different noise levels to assess the learning of aleatoric uncertainty. The test set comprised 200 x-values regularly spaced between $[-40, 40]$ and the corresponding y values were given by $2\sin x$ with $\epsilon = 0$. Consequently, we had three different ranges of x-values to assess epistemic uncertainty. Training region: $[-30, -20]$ and $[20, 30]$, interpolation region: $[-20, 20]$, and extrapolation region: $[-40, -30]$ and $[30, 40]$. We seek to verify that the proposed method can model the aleatoric (different noise levels in the training region) and epistemic uncertainty (interpolation and extrapolation regions).

Ablation study on catalog generation We perform an ablation study to show the effectiveness of tuning both architecture decision variables and training hyperparameters in

AutoDEUQ. First, we designed a high-performing NN by manually tuning the architecture decision variables and hyperparameter configurations on the validation data (see Appendix for the obtained values). We ran AutoDEUQ that used AgEBO for catalog generation and the greedy model selection method for ensemble construction. Next, we used two AutoDEUQ variants: (1) AutoDEUQ (AgE) that used only AgE to explore the search space of the architecture space but used the hand-tuned hyperparameter values following the approach from (Zaidi et al. 2020), and (2) AutoDEUQ (BO) that used the hand-tuned neural architecture and used BO to tune the hyperparameters following the approach from (Wenzel et al. 2021). Finally, we switched off both AgE and BO and trained the manually generated baseline with 500 random weight initializations to build the catalog. All these methods used greedy selection to build an ensemble of size $K = 5$ from their respective catalog of 500 models.

Fig 1 shows the results of these different variants. We can observe that the proposed AutoDEUQ (Fig. 2.a) obtains a superior aleatoric and epistemic uncertainty estimation. The two different noise levels in the training region are well captured by the aleatoric uncertainty estimate. In the interpolation region, aleatoric uncertainty follows the noise levels of the nearby region: for x-values from -20 to 0, it resembles -30 to -20; whereas for 0 to 20, it resembles 20 to 30. The trend is similar for the extrapolation region as well. We can observe that epistemic uncertainty grows as we move from the training data region (grey). Moreover, we can also observe that its magnitude is large for the extrapolation region compared to the interpolation regions. Unlike AutoDEUQ (AgE) and AutoDEUQ (BO), we can see that the epistemic uncertainty grows from $x = -20$, peaks near $x = 0$ and becomes zero near $x = 20$. The results of AutoDEUQ (AgE) and AutoDEUQ (BO) variants are similar: while the aleatoric uncertainty estimates are good, both suffer from poor epistemic uncertainty estimation in the interpolation region. This can be attributed to a lack of model diversity in the ensemble, the former with fixed hyperparameters and the latter with fixed architectures. Finally, the random initialization strategy (Fig. 2.d) with the hand-tuned neural architecture did not model epistemic uncertainty well. This can be attributed to the simplicity of the dataset: given its low dimension, for the same architecture and hyperparameter configurations, the training results in similar NN models.

Comparison between search methods We analyze the impact of different search methods in AutoDEUQ on the uncertainty estimation. We compare the default AutoDEUQ (AgEBO) method (Fig. 1.a) to random search (RS-Mixed) (Fig. 2.a), AgE (AgE-Mixed) (Fig. 2.b), and BO (BO-Mixed) (Fig. 2.c). Note that RS, AgE, BO do not consider the architecture and hyperparameter space separately. Instead, a configuration in the search space is given by a single vector of architecture decision variables and training hyperparameters.

We observe that the uncertainty estimates from the AutoDEUQ (RS-Mixed) are inferior to all other methods. AutoDEUQ (AgEBO) achieves more robust estimates than those of AutoDEUQ (AgE-Mixed) and AutoDEUQ (BO-Mixed). The estimates of epistemic uncertainty for AutoDEUQ

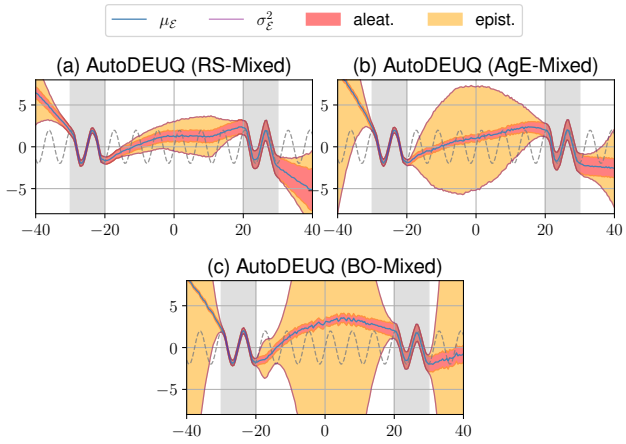

Figure 2: Comparison between different search methods in AutoDEUQ and their impact on uncertainty estimation.

(AgEBO), AutoDEUQ (AgE-Mixed), and AutoDEUQ (BO-Mixed) show a growing trend in the interpolation region as we move away from the training region. AutoDEUQ (BO-Mixed) has larger epistemic uncertainty in the interpolation region than AutoDEUQ (AgEBO) and AutoDEUQ (AgE-Mixed).

The observed differences between the search methods can be attributed to the model diversity in the ensembles. To demonstrate this, we computed the architecture diversity for each method as follows: each architecture was embedded as a vector of integers where each integer represents a choice for one of the decision variable of the neural architecture search space. To compute the diversity of an ensemble, we computed the pairwise Euclidean distance between the embeddings of the architectures composing the ensemble. Then, we kept only the upper triangle of the pairwise distance matrix (because it is symmetric) and normalised it by its norm. Finally, we computed the cumulative sum of the elements of this normalized triangular matrix which gives us a scalar value representing diversity. AutoDEUQ (RS-Mixed) achieved the lowest diversity score (1.41), which also correlates with its poor epistemic uncertainty estimation. While AutoDEUQ (RS-Mixed) obtained diverse models for the catalog, they are not high-performing and consequently the ensemble did not have diverse models. AutoDEUQ (AgE-Mixed) achieved a diversity score of 2.86, which resulted in a better epistemic uncertainty estimate in the interpolation region, but the estimates are poor in the extrapolation region. With a diversity score of 3.49, AutoDEUQ (BO-Mixed) obtained more diverse models, but they contributed to overly large epistemic uncertainty in the interpolation region and extrapolation regions. AutoDEUQ (AgEBO) achieved a diversity score of 3.17, which was in between AutoDEUQ (AgE-Mixed) and AutoDEUQ (BO-Mixed). Moreover, we found that the learning rate values obtained by AutoDEUQ (BO-Mixed) are more diverse than those obtained by AutoDEUQ (AgEBO). The training hyperparameter values obtained by these methods are given in Appendix.

Table 1: Results of the regression benchmark on 10 datasets.

Dataset	NLL						RMSE					
	PBP	MC Dropout	Deep Ens.	Hyper Ens.	DF Ens.	AutoDEUQ	PBP	MC Dropout	Deep Ens.	Hyper Ens.	DF Ens.	AutoDEUQ
boston	2.57	2.46	2.41	**2.15 (0.22)**	2.74	2.46 (0.09)	3.01	2.97	3.28	**2.87 (0.1)**	3.38	3.09 (0.31)
concrete	3.16	3.04	3.06	4.09 (0.17)	3.10	**2.86 (0.07)**	5.67	5.23	6.03	4.7 (0.08)	5.76	**4.38 (0.15)**
energy	2.04	1.99	1.38	0.9 (0.04)	1.62	**0.61 (0.19)**	1.8	1.66	2.09	1.72 (0.08)	2.30	**0.39 (0.02)**
kin8nm	-0.9	-0.95	-1.2	6.89 (2.85)	-1.14	**-1.40 (0.01)**	0.1	0.1	0.09	0.26 (0)	0.09	**0.06 (0.00)**
navalpropulsion	-3.73	-3.8	-5.63	-3.03 (0.49)	-5.73	**-8.24 (0.01)**	0.01	0.01	**0**	0.01 (0)	**0.00**	**0.00 (0.00)**
powerplant	2.84	2.8	2.79	5.24 (0.72)	2.83	**2.66 (0.05)**	4.12	4.02	4.11	4.38 (0.02)	4.10	**3.43 (0.08)**
protein	2.97	2.89	2.83	21.12 (2.52)	3.12	**2.48 (0.03)**	4.73	4.36	4.71	5.09 (0.01)	4.98	**3.52 (0.02)**
wine	0.97	**0.93**	0.94	1.92 (0.92)	1.15	1.00 (0.08)	0.64	**0.62**	0.64	0.73 (0.01)	0.65	**0.62 (0.01)**
yacht	1.63	1.55	1.18	0.48 (0.19)	0.76	**-0.17 (0.11)**	1.02	1.11	1.58	1.86 (0.15)	1.00	**0.44 (0.06)**
yearprediction	3.6	3.59	3.35	7.44 (0.08)	3.58	**3.22 (0.00)**	8.88	8.85	8.89	16.84 (0.08)	9.30	**7.91 (0.04)**
Mean Rank	4.9	3.4	2.5	4.7	3.9	**1.5**	3.7	2.6	3.8	4.6	4	**1.3**

Regression Benchmarks

Here, we compare our AutoDEUQ method with Probabilistic Backpropagation (PBP), Monte Carlo Dropout (MC-Dropout), Deep ensemble (Deep Ens.), distribution-free ensembles (DF-Ens.) and Hyper ensemble (Hyper Ens.) methods. While PBP is selected as a candidate for Bayesian NN, MC-Dropout was selected for its popularity and simplicity. The Deep Ens. (with random initialization of weights, fixed architecture and hyperparameters) will serve as a baseline method. The Hyper Ens. (ensemble with same architecture but with different hyperparameters) is selected because it was a recently proposed high-performing ensemble method.

We used 10 regression benchmark datasets, which were used in the previous literature (Lakshminarayanan, Pritzel, and Blundell 2017; Hernández-Lobato and Adams 2015; Gal and Ghahramani 2016b), to assess the quality of uncertainty quantification methodologies (See Appendix for a description of the datasets). We compare these methods using two metrics: 1) negative log likelihood (NLL) (i.e., how likely are the data to be generated by the predicted normal distribution); and 2) root mean square error metric (RMSE). These two metrics were widely adopted in the literature to compare the quality of uncertainty estimation. The metric values of PBP, MC-Dropout, Deep Ens., and DF-Ens. are copied from their corresponding papers (Hernández-Lobato and Adams 2015; Gal and Ghahramani 2016b; Lakshminarayanan, Pritzel, and Blundell 2017; Pearce et al. 2018), respectively. Nevertheless, we extended and ran the Hyper Ens. method for regression based on the information provided in (Wenzel et al. 2021).

For each dataset, we ran AgEBO to generate a catalog of 500 models and used the *greedy* selection strategy to construct ensembles of $K = 5$ members. We repeated the experiments 10 times with different random seeds for the training/validation split and computed the mean score and its standard error. An exception was the *yearprediction* dataset, which was run only 3 times because the dataset size was large.

The results are shown in Table 1. We can observe that AutoDEUQ obtains superior performance compared to all other methods with respect to both NLL and RMSE. We computed the ranking of the methods for each dataset and computed the mean across the 10 datasets. This is shown in the last row of Table 1. AutoDEUQ with Greedy outperforms all of the other methods on 8 out of 10 datasets. On boston and wine, Hyper Ens. and MC Dropout have the lowest NLL and RMSE values. We note that overall, the recently proposed Hyper Ens. performs worse than all other methods. This can be attributed to the architecture used for regression in Hyper Ensemble, which is a simple MLP network as described in the original paper (Wenzel et al. 2021). This further emphasizes the importance and need for the architecture search for different datasets.

Conclusion and Future Work

We developed AutoDEUQ, an approach to automate the generation of deep ensembles for uncertainty quantification. Ensembling is exploited in uncertainty estimation, and we demonstrated that the predictive variance of the deep ensemble can be decomposed into estimates of aleatoric uncertainty (intrinsic to data) and epistemic uncertainty (captured by model diversity). One important result of this paper is that we empirically demonstrated that epistemic uncertainty is best captured when the models considered in the ensemble are diverse (in hyperparameters and architecture), yet perform all well and similarly on the validation set. This is achieved by a two-step process: (1) using aging evolution and Bayesian optimization to jointly explore the neural architecture and hyperparameter space and generate a diverse catalog of models; (2) using greedy selection of models optimized with the negative log likelihood, to find models that are very different but all with high (and similar) performance.

Using a toy example, we performed an ablation study to visualize the impact of different components of AutoDEUQ on uncertainty estimation. It appears clearly that in regions depleted in training samples, compared to AutoDEUQ, methods optimizing either architecture search or hyper-parameters independently, under-estimate epistemic uncertainty. Finally, we conducted an extensive regression benchmark to compare AutoDEUQ against different classes of UQ methods, involving or not ensembles. Our results confirm quantitatively what was observed on the toy example. The key ingredient of our technique is the diversity and predictive strength and homogeneity of the final ensemble.

Our future work will include (1) applying AutoDEUQ on larger datasets to assess its scalability; (2) evaluating AutoDEUQ on a classification benchmark, and; (3) theoretical insights on the quality of epistemic uncertainty under the various data generation assumptions.

References

Ashukha, A.; Lyzhov, A.; Molchanov, D.; and Vetrov, D. 2020. Pitfalls of in-domain uncertainty estimation and ensembling in deep learning. *arXiv preprint arXiv:2002.06470* .

Bishop, C. M. 1994. Mixture density networks .

Blundell, C.; Cornebise, J.; Kavukcuoglu, K.; and Wierstra, D. 2015. Weight uncertainty in neural network. In *International Conference on Machine Learning*, 1613–1622. PMLR.

Caruana, R.; Munson, A.; and Niculescu-Mizil, A. 2006. Getting the Most Out of Ensemble Selection. In *Sixth International Conference on Data Mining (ICDM'06)*, 828–833. IEEE. doi:10.1109/ICDM.2006.76. URL http://ieeexplore.ieee.org/document/4053111/. ISSN: 1550-4786.

Caruana, R.; Niculescu-Mizil, A.; Crew, G.; and Ksikes, A. 2004. Ensemble selection from libraries of models. In *Twenty-first international conference on Machine learning - ICML '04*, 18. ACM Press. doi:10.1145/1015330.1015432. URL http://portal.acm.org/citation.cfm?doid=1015330.1015432.

Dua, D.; and Graff, C. 2017. UCI Machine Learning Repository. URL http://archive.ics.uci.edu/ml.

Egele, R.; Balaprakash, P.; Vishwanath, V.; Guyon, I.; and Liu, Z. 2021. AgEBO-Tabular: Joint Neural Architecture and Hyperparameter Search with Autotuned Data-Parallel Training for Tabular Data. In *SC21: International Conference for High Performance Computing, Networking, Storage and Analysis*, 1–14. doi:10.1145/3458817.3476203.

Gal, Y.; and Ghahramani, Z. 2016a. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, 1050–1059. PMLR.

Gal, Y.; and Ghahramani, Z. 2016b. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning URL http://arxiv.org/abs/1506.02142.

Ginsbourger, D.; Le Riche, R.; and Carraro, L. 2010. Kriging Is Well-Suited to Parallelize Optimization. In Tenne, Y.; and Goh, C.-K., eds., *Computational Intelligence in Expensive Optimization Problems*, volume 2, 131–162. Springer Berlin Heidelberg. ISBN 978-3-642-10700-9 978-3-642-10701-6. doi:10.1007/978-3-642-10701-6_6. URL http://link.springer.com/10.1007/978-3-642-10701-6_6. Series Title: Adaptation Learning and Optimization.

Hansen, L.; and Salamon, P. 1990. Neural network ensembles 12(10): 993–1001. ISSN 01628828. doi:10.1109/34.58871. URL http://ieeexplore.ieee.org/document/58871/.

Hernández-Lobato, J. M.; and Adams, R. P. 2015. Probabilistic Backpropagation for Scalable Learning of Bayesian Neural Networks URL http://arxiv.org/abs/1502.05336.

Hoffman, M. D.; Blei, D. M.; Wang, C.; and Paisley, J. 2013. Stochastic variational inference. *Journal of Machine Learning Research* 14(5).

Hüllermeier, E.; and Waegeman, W. 2021. Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. *Machine Learning* 110(3): 457–506.

Lakshminarayanan, B.; Pritzel, A.; and Blundell, C. 2017. Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles URL http://arxiv.org/abs/1612.01474.

Maddox, W. J.; Izmailov, P.; Garipov, T.; Vetrov, D. P.; and Wilson, A. G. 2019. A simple baseline for bayesian uncertainty in deep learning. *Advances in Neural Information Processing Systems* 32: 13153–13164.

McLachlan, G. J.; Lee, S. X.; and Rathnayake, S. I. 2019. Finite mixture models. *Annual review of statistics and its application* 6: 355–378.

Mohamed, S.; and Lakshminarayanan, B. 2016. Learning in implicit generative models. *arXiv preprint:1610.03483* .

Neal, R. M. 2012. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media.

Nix, D.; and Weigend, A. 1994. Estimating the mean and variance of the target probability distribution. In *Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94)*, volume 1, 55–60 vol.1. doi:10.1109/ICNN.1994.374138.

Ovadia, Y.; Fertig, E.; Ren, J.; Nado, Z.; Sculley, D.; Nowozin, S.; Dillon, J. V.; Lakshminarayanan, B.; and Snoek, J. 2019. Can you trust your model's uncertainty? Evaluating predictive uncertainty under dataset shift. *arXiv preprint arXiv:1906.02530* .

Pearce, T.; Brintrup, A.; Zaki, M.; and Neely, A. 2018. High-quality prediction intervals for deep learning: A distribution-free, ensembled approach. In *International Conference on Machine Learning*, 4075–4084. PMLR.

Real, E.; Aggarwal, A.; Huang, Y.; and Le, Q. V. 2018. Regularized Evolution for Image Classifier Architecture Search URL http://arxiv.org/abs/1802.01548.

Russell, R. L.; and Reale, C. 2021. Multivariate uncertainty in deep learning. *IEEE Transactions on Neural Networks and Learning Systems* .

Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research* 15(1): 1929–1958.

van Amersfoort, J.; Smith, L.; Jesson, A.; Key, O.; and Gal, Y. 2021. On Feature Collapse and Deep Kernel Learning for Single Forward Pass Uncertainty. *arXiv preprint arXiv:2102.11409* .

Wenzel, F.; Snoek, J.; Tran, D.; and Jenatton, R. 2021. Hyperparameter Ensembles for Robustness and Uncertainty Quantification URL http://arxiv.org/abs/2006.13570.

Wilson, A. G.; and Izmailov, P. 2020. Bayesian deep learning and a probabilistic perspective of generalization. *arXiv preprint arXiv:2002.08791* .

Zaidi, S.; Zela, A.; Elsken, T.; Holmes, C.; Hutter, F.; and Teh, Y. W. 2020. Neural Ensemble Search for Uncertainty Estimation and Dataset Shift. *arXiv preprint arXiv:2006.08573* .

Appendix A: Results

Experimental Settings

We conducted our experiments on the ThetaGPU system at the Argonne Leadership Computing Facility. ThetaGPU is composed of 24 nodes, each composed of 8 NVIDIA A100 GPUs and 2 AMD Rome 64-core CPUs.

For the **generation of a catalog of models** we use different allocations (i.e., number of nodes) depending on the dataset size. During the search, 1 process only using the CPU is allocated for the search algorithm, then neural network configurations (hyperparameters and architecture) are sent to parallel workers for the training. Each worker corresponds to a single GPU. Therefore, for 1 node had 8 parallel workers. For the **construction of an ensemble**, we load all checkpointed models on different GPU instances to perform parallel inferences and then save the predictions to apply the greedy strategy.

On the software side, we used Python 3.8.5 and the core of our dependencies is composed of Tensorflow 2.5.0, Tensorflow-Probability 0.13.0, Ray 1.4.0, Scikit-Learn 0.24.2, Scipy 1.7.0.

Neural Architecture Search

In AutoDEUQ, we used a neural architecture search space of fully-connected like neural networks with possible skip connections. A visualization of this search space is presented in Figure 3. N denotes the number of output variables.

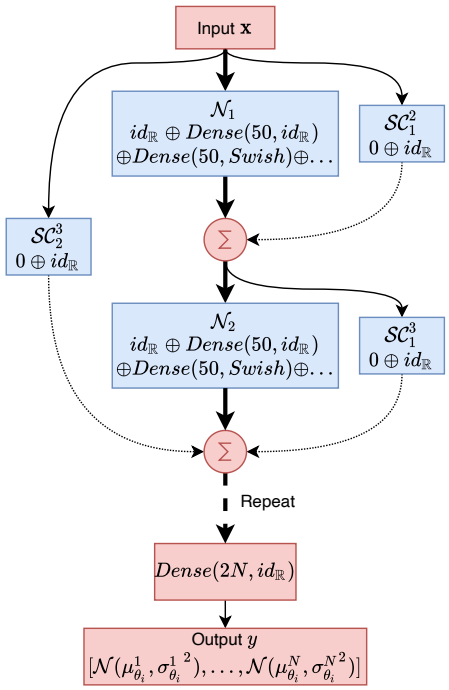

Figure 3: Search space of fully connected neural networks with regression outputs

Toy Example: Ablation study on catalog generation

In the ablation study we use a hand-tuned neural architecture and training hyperparameters. The architecture we used is a two-layer MLP, where each layer has 200 neurons with ReLU activation function. The hyperparameter values obtained were batch size of 16, a learning rate of 0.001, the *Adam* optimizer, a patience of 20 to reduce the learning rate when stagnating, and a patience of 30 to stop training when stagnating.

The Table 2 shows the list of hyperparameters of each member of the ensemble for the different experiments that we conducted.

Toy Example: Comparison between search methods

The Table 3 shows the list of hyperparameters of each member of the ensemble for the different experiments that we conducted.

Benchmark datasets

In Table 4 we give details about the different datasets used in our regression benchmark. These datasets are from the UCI Machine Learning Repository (Dua and Graff 2017).

Table 2: **Ablation study on catalog generation**: training hyperparameter values of the members of the ensemble.

Experiment	Batch Size	Learning Rate	Optimizer	Patience EarlyStopping	Patience ReduceLROnPlateau
Random Initialization	16	0.001	adam	30	20
	16	0.001	adam	30	20
	16	0.001	adam	30	20
AutoDEUQ (AgE)	16	0.001	adam	30	20
	16	0.001	adam	30	20
	16	0.001	adam	30	20
	16	0.001	adam	30	20
	16	0.001	adam	30	20
	16	0.001	adam	30	20
AutoDEUQ (BO)	1	0.000413417	rmsprop	21	14
	1	0.000305604	rmsprop	22	15
	1	0.000413417	rmsprop	21	14
	2	0.000990249	rmsprop	26	19
	1	0.000413417	rmsprop	21	14
	1	0.000442338	rmsprop	21	17
	1	0.000413417	rmsprop	21	14
	1	0.000305604	rmsprop	22	15
	1	0.000413417	rmsprop	21	14
	1	0.000413417	rmsprop	21	14
AutoDEUQ	1	0.000449924	rmsprop	23	15
	1	0.00034571	rmsprop	25	15
	5	0.00459757	adamax	21	14
	1	0.000326253	rmsprop	25	10
	1	0.00034571	rmsprop	25	15
	5	0.00459757	adamax	21	14
	1	0.00034571	rmsprop	25	15

Table 3: **Comparison between joint architecture and hyperparameter search strategies**: training hyperparameter values of the members of the ensemble.

Experiment	Batch Size	Learning Rate	Optimizer	Patience EarlyStopping	Patience ReduceLROnPlateau
RDM-Mixed	3	0.000953022	adam	22	17
	10	0.00102887	nadam	22	17
	3	0.000953022	adam	22	17
AgE-Mixed	3	0.0038734	adamax	22	19
	4	0.00746649	nadam	26	16
	3	0.0038734	adamax	22	13
	4	0.00746649	nadam	26	16
	3	0.0038734	adamax	22	13
	1	0.000471098	rmsprop	30	17
BO-Mixed	1	0.000692455	rmsprop	27	12
	4	0.0006164	adam	21	14
	1	0.000240438	rmsprop	29	15
	1	0.000692455	rmsprop	27	12
	2	0.00134395	rmsprop	20	13
	1	0.000449924	rmsprop	23	15
	1	0.00034571	rmsprop	25	15
	5	0.00459757	adamax	21	14
	1	0.000326253	rmsprop	25	10
	1	0.00034571	rmsprop	25	15
AgEBO	5	0.00459757	adamax	21	14
	1	0.00034571	rmsprop	25	15
	5	0.00459757	adamax	21	14
	1	0.000326253	rmsprop	25	10
	1	0.00034571	rmsprop	25	15
	5	0.00459757	adamax	21	14
	1	0.00034571	rmsprop	25	15

Table 4: Description of the different datasets used in the regression benchmark.

Dataset's Name	Number of Samples	Feature Size
boston	506	13
concrete	1030	8
energy	768	8
kin8nm	8192	8
navalpropulsion	11934	16
powerplant	9568	4
protein	45730	9
wine	1599	11
yacht	308	6
yearprediction	515345	90