Deep Active Learning for regression using ε-weighted Hybrid Query strategy

Harsh Vardhan   HARSH.VARDHAN@VANDERBILT.EDU

Janos Sztipanovits   JANOS.SZTIPANOVITS@VANDERBILT.EDU

Department of Computer Science
Vanderbilt University
Nashville, TN 98195-4322, USA

Abstract

Designing an inexpensive approximate surrogate model that captures the salient features of an expensive high-fidelity behavior is a prevalent approach in design optimization. In recent times, Deep Learning (DL) models are being used as a promising surrogate computational model for engineering problems. However, the main challenge in creating a DL-based surrogate is to simulate/label a large number of design points, which is time-consuming for computationally costly and/or high dimensional engineering problems. In the present work, we propose a novel sampling technique by combining the active learning (AL) method with DL. We call this method *ε-weighted hybrid query strategy (ε-HQS)*, which focuses on the evaluation of the surrogate at each learning iteration and provides an estimate of the failure probability of the surrogate in the Design Space. By reusing already collected training and test data, the learned failure probability guides the next iteration's sampling process to the region of the high probability of failure. We empirically evaluated this method in two different engineering design domains, *finite element based static stress analysis of submarine pressure vessel* (computationally costly process) and second *submarine propeller design* (high dimensional problem). Empirical evaluation shows better performance of our technique as compared to other methods and in one case was able to achieve same level of prediction accuracy as random sample selection process with 58% less number of samples. `https://github.com/vardhah/epsilon_weighted_Hybrid_Query_Strategy`.

Keywords: Deep Learning, Active Learning, Sampling, Propeller design , Finite element analysis.

1. Problem Formulation

In the context of this paper, Design Space Exploration (DSE) is defined as a process to characterize the response surface of a simulator's input–output behavior, where the input describes the relevant system parameters, and the output describes quantities of interest. These simulators can be parameterized and the values of a parameter can be changed in a specified bounded range. These all possible combinations of different parameter values construct the design space. We will first introduce our notations for formalization. Let DS be a bounded subspace of $\mathbb{R}^d$, i.e., $DS \subset \mathbb{R}^d$ and it is represented by an unlabeled set of n samples. d is the number of input parameters that describes the relevant system properties. The training data is $D_{train} = (X, Y^*)$, where X and Y^* are the possible collection of all x_i (input samples to the simulator) and y_i^* (corresponding simulation's output). The role of a simulator is to map the input space to the solution space. i.e. $Sim : X \rightarrow Y^*$. We view these simulators as a nonlinear function which can be modeled with a k–parameter configuration nonlinear regression model i.e. $Sim = Model(\theta_1, \theta_2, ..., \theta_k)$. Accordingly, the surrogate modeling problem is the conditioning of a parametric model (h) on D_{train} to find its parameters (θ) which results into: $h(x, \theta | D_{train}) = y$, $\forall x \in DS : y \approx y^*$ where y^* represents the ground truth. Since our evaluation is costly and/or has high dimensional design space, we only want to take strategic samples with the goal to obtain a certain level of prediction accuracy with the least amount of simulation queries. In

this case, our problem can be formalized as:

$$min\ |D_{train}|$$
$$subject\ to\ \ h(x, \theta|D) = y, \forall x \in DS : y \approx y^*$$

This is an optimization problem, whose direct solution is NP-hard (Cunningham and Schrijver, 1997). However, the heuristic-based iterative approach has been successful and this field of research is known as Active Learning (Settles, 2009). The interpolant model h is an instance of a computational architecture A, which defines the functional dependence of the interpolant on its parameter θ. We choose the architecture A to be a Deep Neural Network(DNN), which has a proven track of powerful learning capability, cheap parallel prediction, and automatic feature extraction. Deployment of active learning strategies is tightly bounded with the class of chosen architecture. Active learning is a semi-supervised machine learning approach for sequential sampling with the goal of identifying strategic samples. The general flow of the AL procedures works as follows:

1. Select some initial randomised samples via methods like Uniform random, Latin hypercube sampling and evaluate these to create $D^{train} \leftarrow D_{train}^{init}$.
2. Train a machine learning model (h) using D^{train} for constraining the sampling procedure in next iteration.
3. Design a *query strategy (Q)* to compute a score ($s \in \mathbb{R}^+$) using the trained model (h) on entire design space (DS).
4. Select new samples in next iteration based on score (s), and evaluate these samples to create new additional training data (D_{train}^{new})
5. Update $D^{train} : D^{train} \leftarrow D_{train}^{new}$
6. Iterate 2-5 until budget of training sample ($B = |D_{train}|$)

By formulating the AL framework in a pool-based sampling scheme, where initially the design space DS has an unlabeled set of n samples and the current labeled training set D^{train} is the labeled set of m samples, our goal is to design a query strategy Q such that $Q : DS \mapsto D^{train}$. The combined goals in a supervised environment can be expressed as follows:

$$\underset{D^{train} \subseteq DS, (x,y) \in D^{train}}{\operatorname{argmin}} \mathbb{E}_{(x,y)}[l(h(x, \theta|D^{train})]$$

$$min\ |D_{train}|$$
$$subject\ to\ \ h(x, \theta|D) = y, \forall x \in DS : y \approx y^*$$

where $l(h(x, \theta|D^{train})$ is the loss function and $l(.) \in R^+$ and the first goal is to reduce loss on D^{train} and the second goal is to reduce m ($m = |D_{train}|$) as small as possible *while ensuring a predetermined level of accuracy on DS*. Therefore, the query strategy $Q : DS \mapsto D^{train}$ is crucial. Also at each acquisition step, we want to select a batch of data points $\{x_1^{sel}, x_2^{sel}, ..., x_b^{sel}\}$ rather than just one point x^{sel} because retraining the learning model (h) on every single additive data-point would be computationally infeasible and even may not give any performance gain. So, based on our current trained model's state $h(w|D_{train})$, we need to design the query strategy (Q) which scores a candidate batch of unlabelled data points ($L_{candidate}$) from DS. An Active learning framework has various query strategies like (uncertainty-based, variance-based, and disagreement-based) however directly applying these strategies to regression problems out of the Bayesian framework is less understood/studied. Accordingly, the standard AL approach is to estimate their uncertainty directly, using Bayesian methods (Rasmussen, 2003), or estimate uncertainty indirectly, through the variance of the outputs of an ensemble of models. Bayesian methods are computationally expensive when the

2

number of samples in DS is more than 10000 and/ or input dimensions are more than 10. Ensemble-like models require independent training of several models, which is also time-consuming due to training a family of models. In this work, we tried to solve this problem out of the Bayesian framework and ensemble models to make it scalable and practically useful for a large class of problems. In the next section, we will explain our approach.

2. Approach

To integrate the DL framework with the AL framework, we propose to use two neural networks in a *student-teacher* fashion, first network is called the *student* network (whose goal is to learn the input-output behavior) and the second network is called the *teacher* network (whose goal is to guide next iteration's sampling process). For training the teacher network, we use already evaluated historical training and test data from prior iterations and label it with a binary indicator (C), where $C : X \mapsto \{0,1\}$, where $C = 1$, indicates student network is able to predict on the sample within an acceptable level of accuracy otherwise $C = 0$. The objective of training the teacher network is to make an inference on the failure probability of the student network on the entire design space. This failure probability can be used to identify the regions/ samples in *DS* on which student network has poor performance/ high likelihood to fail. We continue refining the student and teacher networks on further evaluated samples at each iteration until we exhaust our budget for iterations. (refer figure 1) Ideally, the true failure probability function f^* is unknown but for sampling purposes,
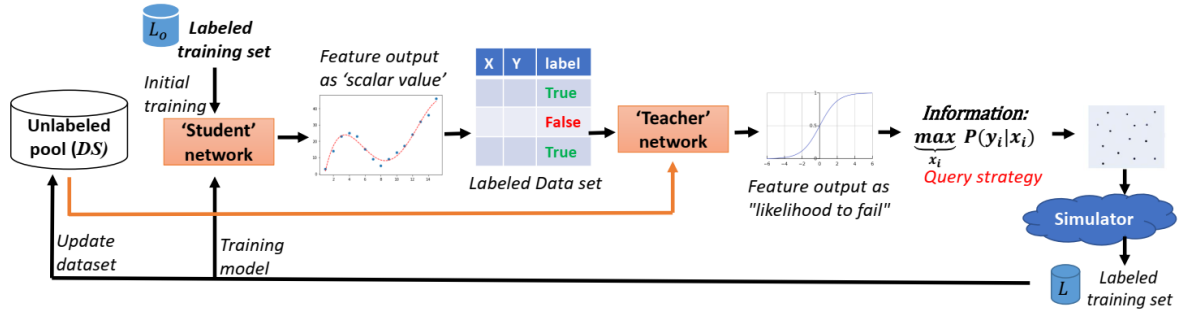

Figure 1: Student teacher architecture: the process

the approximation would work. The benefit of choosing a binary failure indicator is, that we can have a very small DNN because creating a coarse approximation of the decision boundary is a much simpler job than creating a nonlinear regression model. By leveraging the GPU implementation of DNN, we can make inferences on thousands/millions of samples at a cost negligible to running the costly simulation. With the knowledge of the failure probability of student network, it is possible to identify at each iteration, what are the samples $x \in DS$ on which student network has a high likelihood to fail.

2.1 Batching

Using the learned probability function, the active learning framework ideally would select one sample per iteration which has **maximum probability to fail**. However, this one additive sample per iteration does not go well with the DL framework because of retraining time. So, at each iteration, a batch of data points ($L_{candidate}$) should be selected. For a given batch size (B) , the goal is to design an query strategy Q such that $Q : L_{candidate} \rightarrow DS$. The most obvious approach for batching is to select top B samples sorted in descending order by failure probability, we call this approach as **top-B**, where B is the batch size at each iteration. However, this method is highly likely to se-

lect a set of information-rich but similar samples because the query strategy considers each sample independently and ignores the correlation between samples. This may result in a wastage of the evaluation budget. Our goal is not only to select information-rich samples but also to maintain diversity in the selected samples. One way to solve it is by combining probability information with unsupervised clustering. Zhdanov (2019) proposed the batched active learning using the clustering approach and it utilizes both the amount of information in the probability function as well as the diversity of information. We call this approach Diverse Batched Active Learning (**DBAL**).

2.2 Debiasing scheme

Empirical evaluation of both **DBAL** and **top-B** sampling approaches performed poorer than randomized sampling. The underlying reason is, in the initial phase of training, the teacher network which learns the probability function ($f : x \rightarrow [0,1] \: \forall \, x \in DS$) has only limited information about the design space *DS* (a maximum up to training and testing data). Accordingly, the *"trained model does not know what it does not know"* and it creates a biased estimate of the probability function. If f^* is the true failure probability function then f can approximate it only up to $\beta - approximation$. The value of β is an open research question, but we can increase the robustness of the teacher network and can reduce the sensitivity of mismatch between the distribution of f and f^*. For this purpose, we introduce a hyper-parameter ε ($0 < \varepsilon \leq 1$) and adopt filtering of evaluated samples using rejection threshold and then running ε-weighted Monte Carlo sampling on filtered samples. The ε also controls the exploration similar to used in Reinforcement Learning(Sutton and Barto, 2018)(refer algorithm 1). The value of ε reflects our belief in the accuracy of the failure predictor. In the next section, we will empirically evaluate our approach with all above mentioned acquisition functions: **top-B**, **DBAL-10** , **DBAL-50**, ε-**weighted Hybrid Query Strategy**, **Uniformly Random Monte Carlo**.

Algorithm 1 ε-weighted Sampling and de-biasing scheme

1: *Input* : *trained teacher model*(f), *Design space*(DS), *Budget*(B), & ε
2: *Output* : *Candidate Samples*($L_{candidate}$)
3: *Initialization* : $S \leftarrow \{\}$
4: **for** $size(S) \leq \varepsilon * B$ **do**
5:     *Sample proposal instance*/*full DS*, X_t
6:     *Calculate* $f(X_t)$ ▷ Estimate failure probability on samples in DS
7:     $X_f \subseteq X_t : X_f = X_t \: \forall \, X_t$, *where* $f(X_t) \geq 0.5$ ▷ filtered high failure probability samples X_f
8:     $X_{sel} = Choose\_Uniform\_rand(X_f, \varepsilon * B)$ ▷ Choose $\varepsilon * B$ samples randomly from X_f
9:     $S \leftarrow S + X_{sel}$
10: $X_{rest} = Choose\_Uniform\_rand(X_t - X_{sel}, (1-\varepsilon) * B)$
11: $S \leftarrow S + X_{rest}$

---

3. Related Work

There are many ongoing and historical efforts for creating learning-based surrogate models in both propeller design and the FEA domain.(Vesting and Bensow, 2014)(Vardhan et al., 2021)(Liang et al., 2018). However, without any strategic sampling, all work is done for limited design space. Active learning-based sampling by utilizing information-based objective function has also a long history(Fedorov, 2013)(MacKay, 1992)(Settles, 2009) and has been extensively used for regression problems under the Bayesian framework especially by using Kriging and Gaussian Process-based learning models(Forrester et al., 2008)(Rasmussen, 2003). With the advent of deep learning(LeCun et al., 2015), it is found advantageous to integrate DL models(which is a powerful learning model)

with AL to get the best of both worlds however the traditional AL literature has little to offer when the parameters of learning models are large. There is a plethora of ongoing research work in integrating DL with AL and it is called DeepAL but most work is done in the context of classification problems where direct probabilistic output makes creating information-based objective function much easier(Ren et al., 2021). Our work is more inspired by (Uesato et al., 2018)(Vardhan and Sztipanovits, 2021), in which an attempt made to learn the failure probability in the context of rare event failures in RL agents.

4. Experiments and Evaluation

For empirical evaluation, we select two engineering domains–propeller design domain and the Finite Element Analysis (FEA) domain. In both cases, we want to replace a physics-based simulation process with a computationally cheap DNN-based surrogate model. In the propeller domain, the goal is to create a surrogate model to predict the efficiency of the propeller on the given requirement and geometric design. In the case of the FEA domain, the goal is to create a surrogate model of the FEA process to predict the maximum Von mises stress (which determines the integrity of the vessel) in the capsule-shaped pressure vessel used in an Underwater vehicle (refer figures 2 and 3).
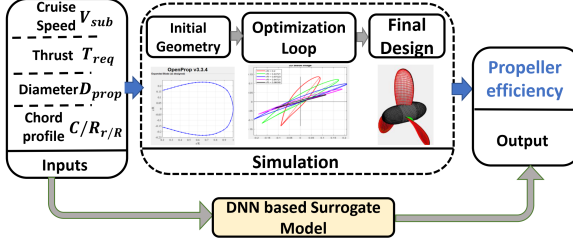

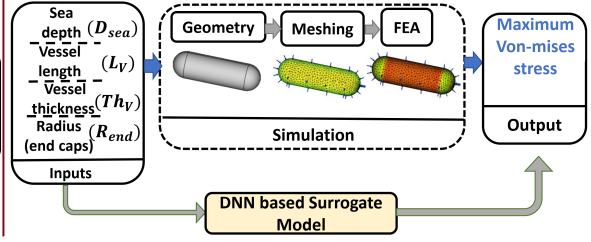Figure 2: Propeller design surrogate

Figure 3: Pressure vessel Surrogate

For data generation and evaluation, we used physics-based simulators -openprop (Epps, Chalfant, Kimball, Techet, Flood, and Chryssostomidis, 2009) and freeCAD (Riegel, Mayer, and van Havre, 2016) for the propeller and FEA domain respectively. In propeller cases, we generated 275000 labeled data in 14 dimensional input parameters space because the average evaluation time per sample was small ($\approx$ 0.5 sec). However, in the case of pressure vessels, the evaluation time is high ($\approx$ 202 sec) and we generated 11131 labeled samples in the 4 dimensional input parameter space. For experiment I in both domains, the total budget of training data is fixed to 2500 (50 iterations of the AL process, with a budget of 50 per iteration). The leftover labeled data is used as the test data (to measure the accuracy of the surrogate). For evaluating the consistency of results, we ran multiple experiments and averaged the prediction accuracy.

We observed that in both domains, **DBAL** and **top-b** strategies perform poorer than the random sampling and the underlying reason is the complete reliance on a biased estimation of failure probability. Figure 6 shows the accuracy of trained DL-based surrogates using different sampling strategies (DL models and other learning parameters are kept the same). Our accuracy metrics is fraction of test predictions which are out of acceptable accuracy(+/- 5% error) i.e., $\frac{\sum_{i=1}^{T} \mathbb{I}(|y_i - y_i^*| \leq 0.05*|y_i^*|)}{T}$. Here y_i is a prediction on i^{th} test sample having ground truth y_i^* and T is the total number of samples in the test data. The **DBAL50** and **DBAL10** are two different DBAL (Diverse Batched Active learning) strategies with hyper-parameter $\beta = 50$ and 10 respectively(refer Zhdanov (2019)). In **random** strategy, at each iteration samples are selected uniformly randomly for unlabeled pool data(DS). Five $\varepsilon - HQS$ approaches are tested labeled as $ep\_g, ep\_025, ep\_05, ep\_075$ and ep_1, called as loga-
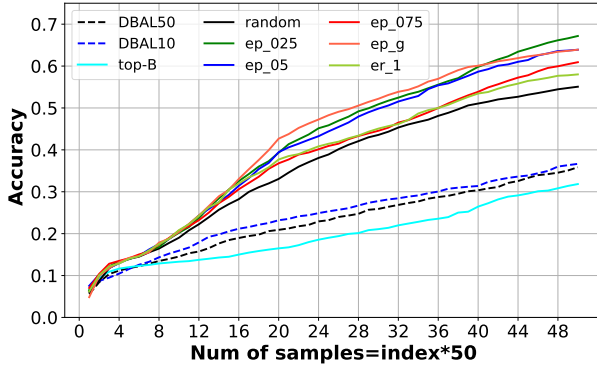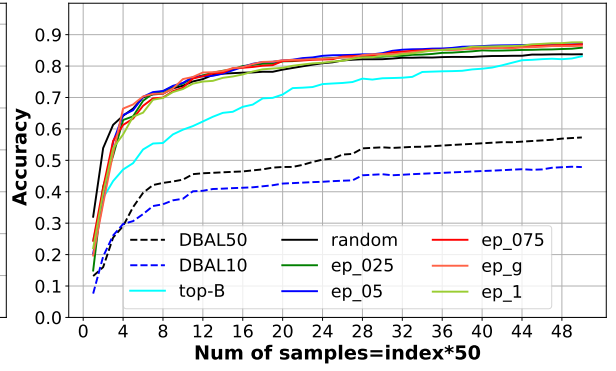
Figure 4: Propeller Domain : Experiment I

Figure 5: FEA domain :Experiment II

rithmic increasing ε, ε with weights 0.25, 0.50, 0.75 and 1.0 respectively. The epsilon-greedy(Sutton and Barto, 2018) approach increases belief in the prediction of teacher network logarithmic, while other fixed ε strategies assign linear belief to the teacher network. For experiment II, we used only FEA domain and reduced the batch size per iteration to 20 and the increased the number of iterations to 400 (i.e., a total budget of training is increased from 2500 to 8000), and the rest of the data is used for testing. We observed the improvement in accuracy of surrogate using *epsilon*-greedy HQS at a higher rate than randomized sampling as the *epsilon*-greedy HQS can find strategic samples, which random sampling cannot. The accuracy increment was 4.14 %(86.47% to 90.61%) for *epsilon*-greedy HQS in comparison to 1.99%(83.76% to 85.75%) in the case of random sampling when training data increased from 2500 samples to 8000 samples. Another result that we want to share is for reaching the same level of accuracy as the randomized sampling strategy, the *epsilon*-greedy HQS saves 57.5% of samples and consequently 250 hours of data generation time (includes the samples simulation time minus training and prediction time from teacher network).

Strategy	Acc-Propeller	Acc-FEA
Random	55.07%	83.76%
Topb	31.83%	83.11%
DBAL10	35.84%	57.28%
DBAL50	36.66%	47.96%
Ep-greedy	63.9%	86.47%
Ep-0.25	67.18%	85.85%
Ep-0.50	63.9%	87.01%
Ep-0.75	60.92%	86.96%
Ep-1.0	58.01%	87.56%

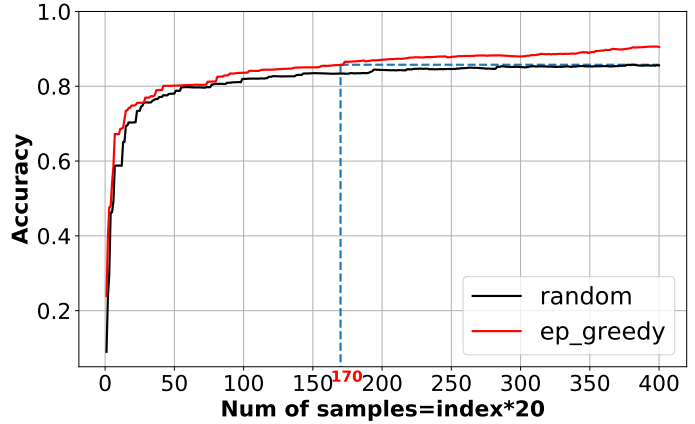Figure 6: Prediction accuracies

Figure 7: FEA domain: Experiment II

5. Conclusion and Future Work

In this work, we developed a new method of data efficient surrogate modeling by combining Deep Learning and Active Learning for regression problem. When compared this methods like DBAL, VMC and top-B strategy, it outperformed all of them.

6. Acknowledgments

References

William J Cook William H Cunningham and William R Pulleyblank Alexander Schrijver. Combinatorial optimization, 1997.

Brenden Epps, Julie Chalfant, Richard Kimball, Alexandra Techet, Kevin Flood, and Chrysssostomos Chryssostomidis. Openprop: An open-source parametric design and analysis tool for propellers. In *Proceedings of the 2009 grand challenges in modeling & simulation conference*, pages 104–111, 2009.

Valerii Vadimovich Fedorov. *Theory of optimal experiments*. Elsevier, 2013.

Alexander Forrester, Andras Sobester, and Andy Keane. *Engineering design via surrogate modelling: a practical guide*. John Wiley & Sons, 2008.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

Liang Liang, Minliang Liu, Caitlin Martin, and Wei Sun. A deep learning approach to estimate stress distribution: a fast and accurate surrogate of finite-element analysis. *Journal of The Royal Society Interface*, 15(138):20170844, 2018.

David JC MacKay. Information-based objective functions for active data selection. *Neural computation*, 4(4):590–604, 1992.

Carl Edward Rasmussen. Gaussian processes in machine learning. In *Summer school on machine learning*, pages 63–71. Springer, 2003.

Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Brij B Gupta, Xiaojiang Chen, and Xin Wang. A survey of deep active learning. *ACM Computing Surveys (CSUR)*, 54(9):1–40, 2021.

Juergen Riegel, Werner Mayer, and Yorik van Havre. Freecad, 2016.

Burr Settles. Active learning literature survey. 2009.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Jonathan Uesato, Ananya Kumar, Csaba Szepesvari, Tom Erez, Avraham Ruderman, Keith Anderson, Nicolas Heess, Pushmeet Kohli, et al. Rigorous agent evaluation: An adversarial approach to uncover catastrophic failures. *arXiv preprint arXiv:1812.01647*, 2018.

Harsh Vardhan and Janos Sztipanovits. Rare event failure test case generation in learning-enabled-controllers. In *2021 6th International Conference on Machine Learning Technologies*, pages 34–40, 2021.

Harsh Vardhan, Peter Volgyesi, and Janos Sztipanovits. Machine learning assisted propeller design. In *Proceedings of the ACM/IEEE 12th International Conference on Cyber-Physical Systems*, pages 227–228, 2021.

Florian Vesting and Rickard E Bensow. On surrogate methods in propeller optimisation. *Ocean engineering*, 88:214–227, 2014.

Fedor Zhdanov. Diverse mini-batch active learning. *arXiv preprint arXiv:1901.05954*, 2019.