ROMANIAN JOURNAL OF INFORMATION SCIENCE AND TECHNOLOGY
Volume **27**, Number 2, 2024, 183–195

On the Modelling Possibilities of Integrated Circuits Behavior Using Active Learning Principles

Vasile GROSU[1,*], Emilian DAVID[3], Liviu GORAS[1, 2], and Georg PELZ[4]

[1] *Gheorghe Asachi* Technical University Iasi, Romania
[2] Institute of Computer Science, Romanian Academy, Iasi Branch, Iasi, Romania
[3] Infineon Technologies Bucharest, Romania
[4] Infineon Technologies Munich, Germany

E-mails: vgrosu@etti.tuiasi.ro*, lgoras@etti.tuiasi.ro, emilianconstantin.david@infineon.com, georg.pelz@infineon.com,
* Corresponding author

Abstract. There are many situations in applications like circuit design, optimization or verification where the simulation time and licensing costs of simulator can become very prohibitive. Therefore, developing metamodels that would mimic circuit behavior for these applications might be highly desired since they can be used at least as a fast preliminary design tool by the engineers to speed up the development process. Efficient sampling strategies can be further employed for further reducing the simulation related costs for designing such metamodels. In this paper we propose two Active Learning sampling schemes that can be used to minimize the number of samples needed for creating reliable metamodels. We validate and compare the approaches with classical fixed sampling schemes on a set of synthetic functions, a simulated circuit and a power device.

Key-words: Active learning; adaptive sampling; integrated circuit design, Gaussian process regression, metamodels; neural networks regression.

1. Introduction

Machine learning (ML) algorithms are used in many applications within various fields among which *computer aided design* (CAD) for *integrated circuits* (IC). Such applications include circuit sizing, design optimization and verification [1–6]. The advantage of modelling the circuit behavior through ML regression models is that of providing a much less expensive solution for evaluating the circuit performances dependencies on circuit parameters and operating conditions

when compared with simulation models regarding time, licensing costs and computational resources. These ML regression models are also referred as metamodels since they mimic the behavior of simulated systems

Traditionally, fixed sampling schemes are used to provide the dataset for building these machine learning models such as *random sampling* (RND) and *Latin hypercube sampling* (LHS), to name a few. The data acquisition process might be a very expensive process, as it may require a large number of simulations for some applications as far as the simulation time for one point can take up to several hours. Thus, it is necessary to explore some more efficient alternatives than the mentioned fixed sampling schemes, in which simulation points are carefully chosen in order to obtain the best model with the least number of points, given a set error threshold or a simulation budget.

Active learning (AL) is an iterative sampling method where the training database is updated based on a selection criterion in order to improve the existing model. Thus, the sampling efficiency is attained through iterations, only the samples that are important in what regards the model accuracy are collected [7–8] compared to the fixed sampling schemes that may acquire unnecessary/redundant data. One way of implementing such a sampling scheme is sampling by the uncertainty/variance of the model [7–8], a lower overall uncertainty meaning and overall better model.

The most natural choice for developing an active learning sampling scheme is by using the *Gaussian process regression* (GPR) since, besides the prediction model, it also provides its uncertainty as the Gaussian process standard deviation function [9]. The training complexity of GPR scales cubically with training samples number [9], making this implementation unfeasible for larger datasets. In this sense, other alternatives have to be considered for the case when training time becomes too prohibitive. In this paper we consider two AL sampling approaches, one based on GPR (ALS-GP) and the other on neural network (ALS-NN) regression [10], and compare their performances in modelling circuits and functions of various complexities.

The remainder of this paper is organized as follows: Section 2 presents the general framework used for sampling the search space and does an introduction to the regression algorithms used in the paper. Section 3 presents a qualitative and quantitative comparison of the obtained results when using the two proposed AL sampling schemes while also taking into account some fixed sampling schemes.

The remainder of this paper is organized as follows: Section 2 presents the general framework used for sampling the search space and does an introduction to the regression algorithms used in the paper. Section 3 presents a qualitative and quantitative comparison of the obtained results when using the two proposed AL sampling schemes while also taking into account some fixed sampling schemes.

2. Active Learning Sampling

Active learning sampling is a technique used in various fields such as statistics, signal processing, and computer graphics, to improve the efficiency of data gathering by dynamically adjusting the sampling process based on the characteristics of the data being collected [7–8]. The basic idea behind AL sampling is to focus data gathering efforts on regions of interest where data is more likely to contain valuable information, and reduce sampling in regions where data is unlikely to provide much insight. This can result in significant improvements in the quality and efficiency of data collection, especially in situations where the cost or difficulty of collecting

data is high. One of the key advantages of AL sampling is that it can reduce the overall amount of data that needs to be collected, while still maintaining the desired level of accuracy.

Fig. 1 presents the general framework of the AL sampling method, where the *regression* stage can be represented in general by any ML regression. The GPR is the preferred one as it has advantage of naturally providing a regression uncertainty function in the form of Gaussian process standard deviation that can be used directly in the AL sampling scheme, as previously reported in [11]. If for various reasons other type of regression is used, additional effort must be spent in order to obtain the regression uncertainty estimate.

The AL sampling process starts with a set of initial samples that are obtain through circuit simulation. The amount of necessary initial samples can be debatable and we have used in our experiments $10n$, where n is the number of input parameters. However, this number can be chosen through many other considerations depending on the application requirements [8]. The AL sampling scheme will iteratively add new training simulation samples on the regression maximum uncertainty location until the stopping criterion is met. For example, the stopping criterion can be a given simulation budget or minimal regression error.

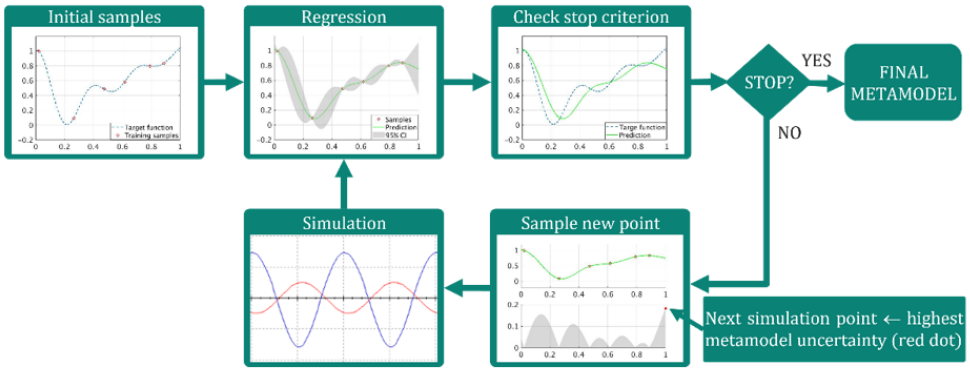

Fig. 1. General framework of the active sampling approach.

2.1. Gaussian process regression

Gaussian process regression is a supervised learning algorithm [9], inspired from Kriging that it is used to fit a function to represent the training data points in order make predictions at new data points. Given a set of N data point pairs $\mathbf{X} = [\mathbf{x_1} \dots \mathbf{x_i} \dots \mathbf{x_N}]$ and $\mathbf{y} = [y_1 \dots y_i \dots y_N]$ generated by the unknown function $f(\mathbf{x})$ as:

$$y = f(x) + \varepsilon,\ x \in R^n,\ y \in R,\ \varepsilon \in N(0,\ \sigma_n^2) \tag{1}$$

the GPR model makes assumption about this data as being possibly generated by functions from an infinite set of functions. This collection of functions is characterized by a Gaussian distribution with the mean $m(x)$ and variance σ^2, given the initial sample values:

$$f^*(x)\,|X,\ y,\ x \sim N\left(m\left(x\right), \sigma^2\left(x\right)\right) \tag{2}$$

$$m\left(\mathbf{x}\right) = k\left(\mathbf{x}, \mathbf{X}\right)^T \left[K_N + \ \sigma_n I\right]^{-1} \mathbf{y} \tag{3}$$

$$\sigma^2\left(\mathbf{x}\right) = k\left(\mathbf{x}, \mathbf{x}\right) - k\left(x, \mathbf{X}\right)^T \left[K_N + \sigma_n I\right]^{-1} k(x, \mathbf{X}) \tag{4}$$

where K_N is the covariance matrix defined as:

$$K_N = \begin{bmatrix} k_{11} & k_{12} & \cdots & k_{1j} \\ k_{21} & k_{22} & \cdots & k_{2j} \\ \vdots & \vdots & \ddots & \vdots \\ k_{i1} & k_{i2} & \cdots & k_{ij} \end{bmatrix} \tag{5}$$

defined by the kernel function that can be a Gaussian function:

$$k_{ij} = k\left(\mathbf{x_i}, \mathbf{x_j}\right) = \ \sigma_p^2 e^{\frac{(||\mathbf{x_i} - \mathbf{x_j}||)^2}{2l^2}} \tag{6}$$

where σ_p and l are the scaling factor and the Gaussian width. Together with noise standard deviation σ_n they are commonly referred as model hyperparameters [9]. In our experiments we considered that the model hyperparameters were chosen by maximizing the marginal log likelihood as presented in [5].

It can be easily observed that given a GPR model the prediction can be performed according to $m(x)$, while $\sigma^2(x)$ can be considered as model uncertainty. Naturally, this uncertainty measure can be used as an indicator about where the generating function has to be sampled in case we want to improve the model accuracy by acquiring new data (Fig. 1). Thus, an AL scheme can be easily developed by using the GPR in a loop where with each iteration a new data point is acquired at maximum uncertainty locations given by:

$$x_{\text{next}} = \text{argmax}\left(\sigma^2\left(\mathbf{x}\right)\right) \tag{7}$$

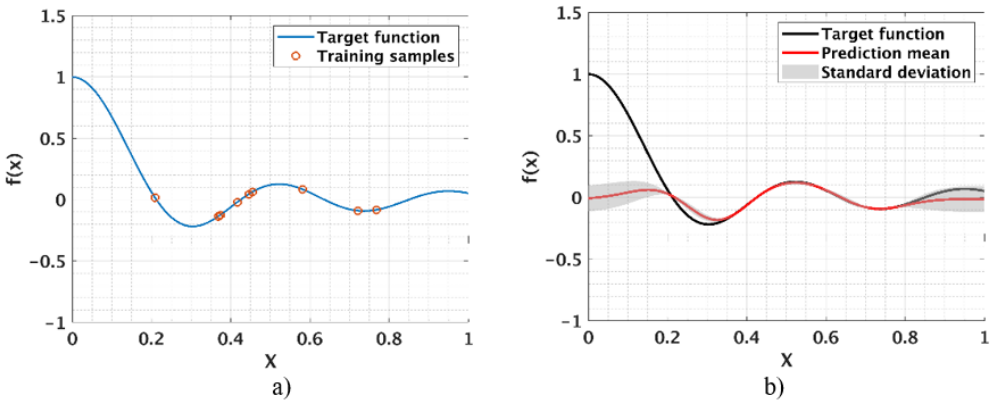

Fig. 2. Training and prediction of GPR: a) Target function and training samples; b) Prediction of GPR post training with the samples in a).

This active learning approach can be considered as a Bayesian optimization framework that works with *upper confidence bound* (UCB) as the acquisition function [9], where the model mean contribution is ignored:

$$UCB\left(\mathbf{x}\right) = m\left(\mathbf{x}\right) + k\sigma^2\left(\mathbf{x}\right) \tag{8}$$

Several papers have successfully used GPR or similar approaches within active learning strategies in order to efficiently improve the prediction accuracy [3–7]. GPR is an attractive candidate for this kind of application because it naturally provides the prediction uncertainty. Unfortunately, since GPR doesn't scale well with the amount of training data, other algorithms such as neural networks [10] could be used but additional effort has to be put in deriving the model uncertainty.

2.2. Active learning implementation using neural network regression

The input-output relationship of a multilayer perceptron (MLP) artificial neural network [10] can be expressed as:

$$F\left(\mathbf{x}\right) = \sum_{i=1}^{m} \alpha_i \tanh(\mathbf{W}^T\mathbf{x} + \mathbf{b}) \tag{9}$$

and might be represented schematically as shown in Fig. 3 where the input is a vector of n variables (features) $\mathbf{x} \in \mathbb{R}^n$ with $\mathbf{x} = [\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n}]$ and the output is a scalar variable $F(x) \in \mathbb{R}$. α_i are the output weights that connect the hidden layer to the output and $\mathbf{b} \in \mathbb{R}^m$ is the bias vector. The matrix $\mathbf{W} = [\mathbf{w}_1 \ldots \mathbf{w} \ldots \mathbf{w}_m]$ is a n-by-m matrix that connects the inputs to the hidden layer of m neurons characterized by a hyperbolic tangent function as the activation function.

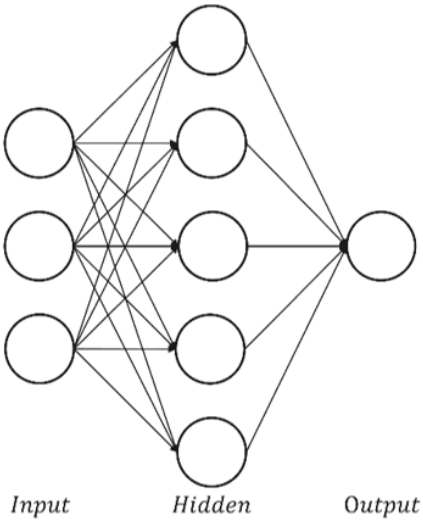

Fig. 3. The schematic representation of an artificial NN.

The main property of MLP NNs of the form (9) is that they are universal approximators [10], being capable to model any continuously differentiable unknown function over a given interval. The extension of the MLP to multiple hidden layers is possible but for the single purpose of function approximation, one hidden layer is sufficient. The process of adjusting the network free parameters W, α, b in order approximate a function $f(x)$ is commonly referred as training procedure. Thus, for training the NN we use a set of M data point pairs $\mathbf{x} = [\mathbf{x_1} \dots \mathbf{x} \dots \mathbf{x_M}]$ and $\mathbf{y} = [_1 \dots \dots _M]$ generated by the unknown function $f(x)$ (simulator in our case) as:

$$\mathbf{y} = f(s) \tag{10}$$

The neural network has the advantage of a better training algorithm scaling with the training data quantity when compared to GPR especially when fast converging training algorithm are used like Levenberg-Marquardt [12]. Unlike GPR, the NN does not provide a regression uncertainty function thus, additional effort needs to be spent to derive one.

The uncertainty function can be estimated in the NN case as the standard deviation of multiple regression models, each being trained with partially overlapping training sets. In our experiments we have used 10 neural networks that were trained using a 10-fold training/validation procedure [13]. These models are used for deriving their mean:

$$m(\mathbf{x}) = \frac{1}{L} \sum_{i=1}^{L} f_i(\mathbf{x}) \tag{11}$$

and variance:

$$\mathbb{V}(\mathbf{x}) = \frac{1}{L} \sum_{i=1}^{L} (f_i(\mathbf{x}) - m(\mathbf{x}))^2 \tag{12}$$

They are considered the equivalent of the GPR mean and variance and as a consequence they can be used in the AL algorithm accordingly. Fig. 4.a depicts the neural networks regressions trained as previously mentioned and Fig. 4.b represents their corresponding mean and variance estimated using (11) and (12). The regression modelling set up is similar to that presented in Fig. 2 in order to put in evidence the similarity of the two approaches results.

While creating multiple NN regressions instead of single GPR may seem more computational expensive at first glance, this is not the case, especially in the case of higher input variable dimension and/or large training data set.

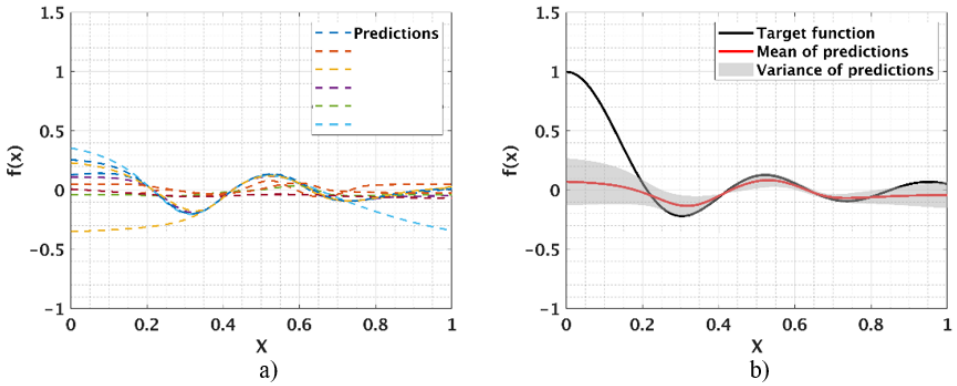

Fig. 4. Emulating the outputs of GPR with NN's: a) Predictions of several NN's; b) Prediction of variance and mean from functions in a).

Table 1. Training time table (in seconds)

	n=1		n=3		n=7		n=11		n=17	
	NN	GP	NN	GP	NN	GP	NN	GP	NN	GP
100	1	2	1	3	2	3	2	2	2	9
200	1	6	1	6	3	6	2	4	2	11
1000	1	64	2	50	5	60	3	76	3	84
2000	3	57	4	67	8	67	4	82	5	88
10000	5	181	14	137	47	194	32	239	24	211
20000	12	300	26	301	66	486	67	503	63	630

Table 1 presents a comparative analysis of the training time required for creating NN and GP regressions give the same set of training data of various sample size that was generated by a function with arbitrary input variable dimension. We can observe that training a neural network is much faster than training a GPR. Hence, even if we use several models to derive the uncertainty metric, we can still reduce the overall time needed to sample new points. For training the regression models we have used MATLAB's Statistics and Machine Learning Toolbox [14]. The presented results were obtained on training a surrogate function from [15] that can be easily extended to arbitrary input variable dimension. This function is presented in Fig. 5 in the 2D case.

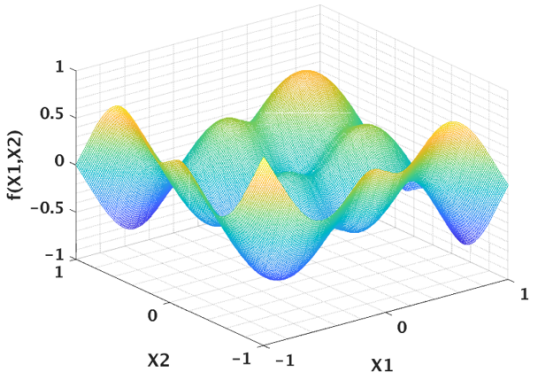

Fig. 5. F21 from [15] – 2D example.

3. Experimental Results

In this section we will assess the capability of our AL approaches in modelling the circuit's behavior optimally in what regards the needed number of simulations, when compared with fixed sampling schemes. In our experiments, all the resulting ML regression models quality will be assessed using the relative root mean square error ($rRMSE$):

$$rRMSE = \frac{\sqrt{\sum_{i=1}^{n} \left(yt_i - m\left(\mathbf{xt}_i\right)\right)^2}}{\sqrt{\sum_{i=1}^{n} \left(yt_i\right)^2}} \tag{13}$$

where yt_i are the test output samples and $m(\mathbf{xt}_i)$ are their corresponding mean predictions of the regression algorithm used for the values in the test input variables $\mathbf{xt}_i$. In our experiments we have considered an independent test set of $n = 1000$ samples, but in practice this number can be also limited by the simulation costs. Also, all involved variables, inputs or outputs are normalized to unity hypercube before creating the regression models.

In our experiments we create regression models using an increasingly number of samples that use various sampling schemes, some being fixed (RND and LHS) some using AL sampling schemes (ALS-GP, ALS-NN). All the regression models are represented by GPR except the ALS-NN ones. For this case, the model is represented by the best validation error neural network from the ten we train for creating the model statistics from (11) and (12).

3.1. Comparison of sampling algorithms for synthetic functions behavior modelling

When working with time expensive simulations it is a good practice to first develop and test the ML algorithm on a set of mathematical/surrogate functions. Besides the rapidity to evaluate advantage, they also offer full control regarding their complexity and input dimensionality. Thus, it is possible to have a more complete overview of the AL behavior than what real circuits simulations may offer due to their limited availability and behavior diversity choice. In Fig. 6 (a) and (b) we can observe the regressions errors trend that use training samples acquired by the various sampling schemes for the function presented in Fig. 5 for the 2D case.

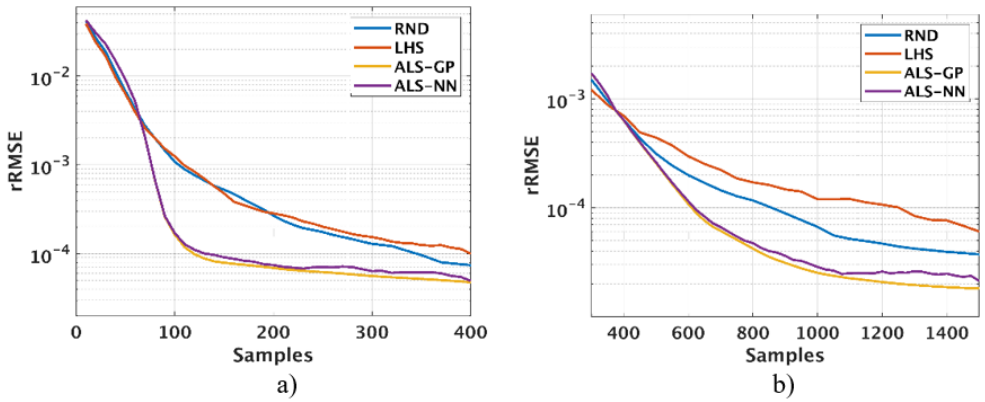

Fig. 6. Error trend for F21: a) in the 2D case; b) in the 3D case.

It is obvious that the AL sampling scheme has the best overall error trend when compared to the fixed sampling methods. Furthermore, there is minimal loss in accuracy between the GPR and NN regression approaches. The fact that the error trends are not strictly monotonously decreasing can be partly explained by the stochastic nature of learning in general [9, 10]. For the particular case of GP regression this may come from hyperparameter optimization procedure [9] which introduces a mild stochastic component in the test error behavior. This happens despite the fact that the GP model is the same given same data and same hyperparameters. For the NN regression case the non-monotonous behavior of the error trend is more pronounced as it is well known that the training algorithm may stuck in different local minima due to random network parameter initialization [10]. NN hyperparameter optimization may further explain this effect.

3.2. Comparison of sampling algorithms for integrated circuit behavior modelling

We have applied the presented sampling schemes for modelling the performance behavior for two simulated device and integrated circuit use cases: a discrete device represented by Infineon Power MOSFET Switch [16] and an Infineon Proprietary LDO [17]. The choice of these use cases was guided by the fact that their simulation can be expensive in their own way and to prove that it can be used for a certain variety of IC design and optimization use cases and simulation types.

3.2.1. Power switch case

The Power MOSFET Switch behavior is characterized by physical level steady state simulation using *technology computer aided design* (TCAD) tools like COMSOL [18]. This simulation type is quite computational and time demanding as it revolves in numerically solving a huge set of partial derivative nonlinear equations that take into account possible intricate device geometries and material (physical, electrical, thermal) properties. Depending on the simulation setting and device complexity one simulation time may take between several hours to even days. Traditionally, during design phase engineers vary the device design parameters (geometries, physical

properties) in order to obtain the desired electrical properties (R_{on} resistance) and if the process is successful, the device is considered for other design, verification and production steps.

Very often, it happens that various other devices with other electrical properties need to be produced using the same technology that are different from the initial device design only in what regards the geometrical parameters. In this case, the process of device optimization is repeated and further simulations need to be performed, leading to huge costs in what regards time and software licenses. Our methods can be extremely useful in overcoming this issue since they can automatically model (characterize) the device behavior (*e.g.* electrical properties) on the design parameters variation, optimally in what regards the necessary number of simulations. This ML model can be further used for redesign or other optimization purposes without the need of reiterating the simulations. In our experiments we have used a simulated Power MOSFET Switch where we have modeled the on-state resistance (R_{on}) dependence on its geometric properties (chip width/length and clip width/length) [18, 19].

For the power switch R_{on} resistance is one of the most important parameters thus having an accurate model of it can help to optimize or predict if certain design is feasible. Similar with the synthetic functions case, Fig. 7 show that the AL sampling methods provides the best results in what regards the error of the device regression model.

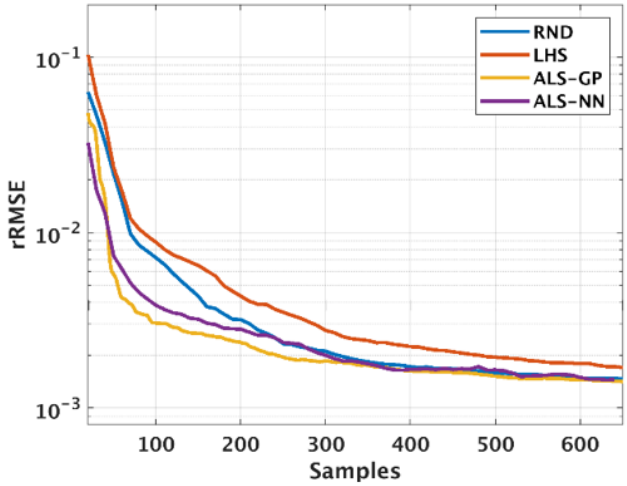

Fig. 7. Error trend for the modelled R_{on} of the 4 factor MOSFET Switch.

3.2.2. Low dropout voltage regulator

The other use case refers to an LDO design where the situation can be the same in principle. A general description of such IC is presented in [17]. The circuit design is performed in order to fulfill certain product specifications in what regards its electrical performances. These *electrical performances* (EP) are verified in the pre-silicon development phase using SPICE simulations [20] that can also be time consuming, as they use very complicated and accurate device models libraries. There is usually the case when a variety of products needs to be produced having the same base design but with different specifications. It is also be possible that a given IC design to be the subject of further optimization. This is usually done by modifying the design and de-

vices parameters and performing more simulation in order to verify that optimization targets are fulfilled. Our AL methods can be used for reducing the simulation costs associated with the optimization iterations by efficiently modelling the IC behavior on the important design parameters. The resulting regression model can be used for the required IC optimization iterations without the need to perform simulations in the loop. In our experiments we have tested the AL sampling performances for an LDO use case where its EP are characterized by the *power supply rejection rate* (PSRR) that is influenced by 6 design parameters (circuit resistances, capacitances, transistors widths and lengths).

For the LDO Voltage regulator use case we have considered the average *power supply rejection rate* (PSRR) [17]. Fig. 8 represents the general LDO topology that indicates more specifically on which design parameters we are referring. The real design architecture we have used is Infineon Proprietary which we cannot disclose.

As it can be observed in Fig. 9, the results for the modelled PSSR are consistent with the previously presented use case and surrogate modelling case. The general conclusion is that the AL sampling schemes are more accurate given any number of variable samples used in training.

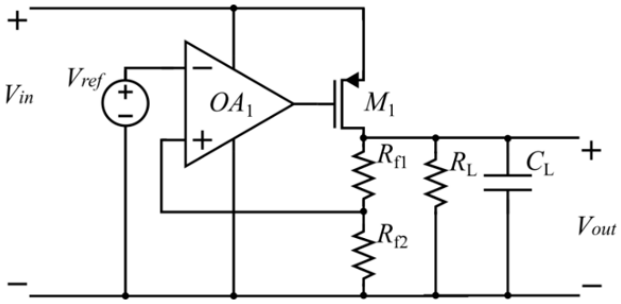

Fig. 8. Generic LDO topology.

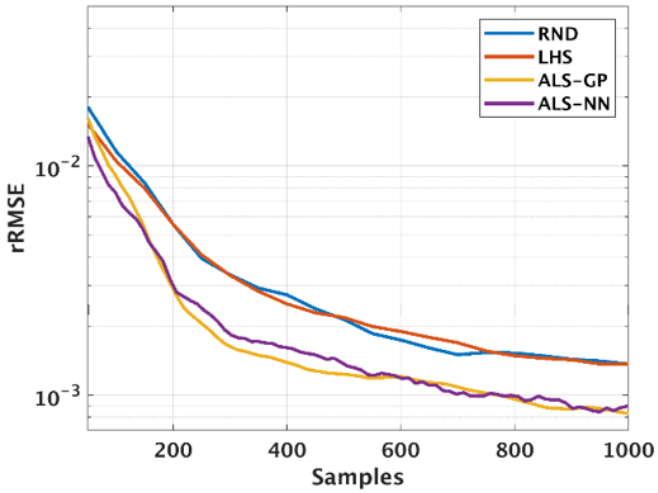

Fig. 9. Error trend for the modelled PSRR of the 6 parameters LDO.

4. Conclusions

In this paper we have presented two efficient AL based sampling algorithm that can be used to model circuits or devices behavior by the means of ML regressions in what regards their performance dependence on the design parameters. The main benefit of the presented methods consists in reducing the costs associated with the necessary simulations that are needed to characterize IC behavior. The resulting ML model can be used for further IC related applications that involves design optimization, circuit sizing or performances tuning.

Even if the GPR based AL method seem to be more natural and accurate, we have provided also a similar NN base alternative in order to circumvent the GPR limitations in what regards data scaling.

It was shown that the use of the AL sampling methods for creating IC metamodels has strong advantages regarding the data efficiency when compared to the fixed sampling methods. This was experimentally proven on a diverse set of use cases that involved synthetic functions, a physically simulated power device and a simulated IC design.

As further research, we aim at incorporating and testing our approach on more specific IC design applications like circuit optimization and sizing or IC verification.

References

[1] J. HUANG, S. ZHANG, C. TAO, G. YANG, C. YAN, D. ZHOU and X. ZENG, *Bayesian optimization approach for analog circuit design using multi-task Gaussian process*, Proceedings of 2021 International symposium on Circuits and Systems, Daegu, Korea, 2021, pp. 1–5.

[2] C. VIŞAN, M. SIEBERER and H. CUCU, *Designer-like automated circuit sizing for multiloop LDO*, Proceedings of 2023 International Semiconductor Conference, Sinaia, Romania, 2023, pp. 103–106.

[3] B. HE, S. ZHANG, F. YANG, C. YAN, D. ZHOU and X. ZENG, *An efficient Bayesian optimization approach for analog circuit synthesis via sparse Gaussian process modeling*, Proceedings of 2020 Design, Automation & Test in Europe Conference & Exhibition, Grenoble, France, 2020, pp. 67–72.

[4] A. C. SANABRIA-BORBÓN, S. SOTO-AGUILAR, J. ESTRADA-LÓPEZ, D. ALLAIRE and E. SÁNCHEZ-SINENCIO, *Gaussian-process-based surrogate for optimization-aided and process-variations-aware analog circuit design*, Electronics **9**(4), 2020, paper 685.

[5] T. NGUYEN and J. SCHUTT-AINE, *Gaussian orocess surrogate model for variability analysis of RF circuits*, Proceeding of 2020 IEEE Electrical Design of Advanced Packaging and Systems, Shenzhen, China, 2020, pp. 1–3.

[6] M. WANG, W. LV, F. YANG, C. YAN, W. CAI, D. ZHOU and X. ZENG, *Efficient yield optimization for analog and SRAM circuits via Gaussian process regression and adaptive yield estimation*, IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems **37**(10), 2018, pp. 1929–1942.

[7] J. FUHG, A. FAU and U. NACKENHORST, *State-of-the-art and comparative review of adaptive sampling methods for kriging*, Archives of Computational Methods in Engineering **28**, 2021, pp. 2689–2747.

[8] L. HAITAO, O. YEW-SOON and C. JIANFEI, *A survey of adaptive sampling for global metamodeling in support of simulation-based complex engineering design*, Structural and Multidisciplinary Optimization **57**, 2018, pp. 393–416.

[9] C. E. RASMUSSEN and C. WILLIAMS, *Gaussian Processes for Machine Learning*, MIT Press, Cambridge, MA, 2006.

[10] S. HAYKIN, *Neural Networks and Learning Machines*, Pearson Education India, Noida, 2009.

[11] V. GROSU, E. DAVID, L. GORAS and G. PELZ, *Modelling integrated circuit behavior using an active learning approach based on Gaussian process regression*, Proceedings of 2023 International Semiconductor Conference, Sinaia, Romania, 2023, pp. 245–248.

[12] D. W. MARQUARDT, *An algorithm for least-squares estimation of nonlinear parameters*, Journal of the Society for Industrial and Applied Mathematics **11**(2), 1963, pp. 431–441.

[13] E. J. DE FORTUNY and D. MARTENS, *Active learning-based pedagogical rule extraction*, IEEE Transactions on Neural Networks and Learning Systems **26**(11), 2015, pp. 2664–2677.

[14] M. PALUSZEK and S. THOMAS, *Practical MATLAB Deep Learning*, Apress, Berkeley, CA, 2020.

[15] V. PLEVRIS and G. SOLORZANO, *A collection of 30 multidimensional functions for global optimization benchmarking*, Data **7**(4), 2022, paper 46.

[16] S. KRISHNA, *The figure of merit of a semiconductor power electronics switch*, IEEE Transactions on Electron Devices **65**(10), 2018, pp. 4216–4224.

[17] G. A. RINCÓN-MORA, *Analog IC Design with Low-Dropout Regulators (LDOs)*, McGraw-Hill, Englewood Cliffs, NJ, 2009.

[18] Semiconductor Module User's Guide, Version 5.3a, COMSOL Multphysics, Stockholm, Sweden, 2017.

[19] L. CHEN and H. BAGCI, *Steady-state simulation of semiconductor devices using discontinuous Galerkin methods*, IEEE Access **8**, 2020, pp. 16203–16215.

[20] K. KUNDERT, *The Designer's Guide to SPICE and SPECTRE*, Springer, Berlin, Heidelberg, New York, 2006.