CALT: *C*lassification with *A*daptive *L*abeling *T*hresholds for Analog Circuit Sizing

Zhengfeng Wu
Drexel University
Philadelphia, PA
jeff.wu@drexel.edu

Ioannis Savidis
Drexel University
Philadelphia, PA
isavidis@coe.drexel.edu

ABSTRACT

A novel simulation-based framework that applies classification with adaptive labeling thresholds (*CALT*) is developed that auto-generates the component sizes of an analog integrated circuit. Classifiers are applied to predict whether the target specifications are satisfied. To address the lack of data points with positive labels due to the large dimensionality of the parameter space, the labeling threshold is adaptively set to a certain percentile of the distribution of a given circuit performance metric in the dataset. Random forest classifiers are executed for surrogate prediction modeling that provide a ranking of the design parameters. For each iteration of the simulation loop, optimization is utilized to determine new query points. *CALT* is applied to the design of a low noise amplifier (LNA) in a 65 nm technology. Qualified design solutions are generated for two sets of specifications with an average execution of 4 and 17 iterations of the optimization loop, which require an average of 1287 and 2190 simulation samples, and an average execution time of 5.4 hours and 23.2 hours, respectively. *CALT* is a specification-driven design framework to automate the sizing of the components (transistors, capacitors, inductors, etc.) of an analog circuit. *CALT* generates interpretable models and achieves high sample efficiency without requiring the use of prior circuit models.

CCS CONCEPTS

• **Hardware→Analog and mixed-signal circuit optimization**; **Methodologies for EDA**; • **Computing methodologies**→ *Supervised learning by classification*.

ACM Reference Format:
Zhengfeng Wu and Ioannis Savidis. 2020. *CALT* : *C*lassification with *A*daptive *L*abeling *T*hresholds for Analog Circuit Sizing. In *2020 ACM/IEEE Workshop on Machine Learning for CAD (MLCAD '20), November 16–20, 2020, Virtual Event, Iceland.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3380446.3430633

1 INTRODUCTION & BACKGROUND

Traditionally, the design of an analog integrated circuit is completed by solving analytic equations that link design parameters with performance metrics. To automate the sizing of the components (transistors, capacitors, inductors, etc.) of an analog circuit, multi-objective optimization problems are formulated with analytic equations [1][2]. The generated Pareto fronts provide a means to analyze the tradeoffs in circuit performance. However, with technology scaling, the knowledge-based approaches are limited by the increasing complexity of circuit equations. In addition, extra effort is required to tune the circuit to resolve any mismatch between theoretically optimized results and simulation results.

Simulation-based approaches emerge as a substitute that addresses the challenges associated with knowledge-based optimization methods. Data mining and machine learning techniques are utilized to extract modeling and design information from simulation data in a bottom-up approach. Representative techniques include stochastic pattern search [3], Bayesian optimization [4], and deep neural networks [5]. Prior work has shown that simulation-based methods are successful in the design of analog circuits. However, improvements are needed with regard to:

- **Sample efficiency**: Simulation-based methods rely on real-time sampling and optimization with simulation tools. The slow numerical solvers used for simulation limit the size of the dataset. To improve sample efficiency, a technique that samples from high-dimensional black-box functions with Duchon pseudo-cubic splines is proposed in [6]. Bayesian neural networks are described in [7] to approximate the Pareto front with a reduced number of samples. Reducing the number of samples required by the optimization process to shorten the design time remains an open challenge.
- **Specification-driven design considerations**: Based on the circuit requirements, analog design specifications are grouped into two categories: 1) **figure of merit (*FoM*) constraints** that require optimization, and 2) **hard constraints** that must only be sufficiently met. As an example, power is treated as an *FoM* constraint when a design priority is to minimize the power consumption. In contrast, power consumption is treated as a hard constraint, specifically a power budget, when other circuit metrics are more critical.

  FoM constraints are commonly optimized by regression models [5][7]. In practice, a limited number of circuit performance metrics are considered *FoM* constraints, for two primary reasons. First, when less important metrics are overemphasized, the search space is narrowed unnecessarily, which results in a more difficult or even infeasible search. Second, when more than two metrics are concurrently considered as *FoM* constraints, the Pareto fronts generated by multi-objective optimization algorithms such as *NSGA II* [8] are hard to visualize and apply. Tradeoff curves between two circuit metrics are meaningful only when the remaining specification-based metrics are satisfied.

  In practice, specifications are often listed in the form of hard constraints, where the objective is to meet the set of target values. Applying classification to predict whether a candidate design point satisfies the specifications is well suited for analysis with hard constraints. In [9], support vector machines (SVMs) are introduced to classify the performance space of analog circuits. One-class classifiers are favored over two-class classifiers as the latter suffers from a large dimensionality of the parameter space. Specifically, the proportion

MLCAD '20, November 16–20, 2020, Virtual Event, Iceland

ACM ISBN 978-1-4503-7519-1/20/11…$15.00
https://doi.org/10.1145/3380446.3430633

of design points that yield the desired performance parameters is likely to be small in an initial randomly sampled dataset. The dimensionality of the design space, therefore, limits the application of binary classifiers. Additionally, in [9], classifiers are only applied for the analysis of the circuit performance space rather than for the design of the circuit.

- **Interpretability**: Ideally, techniques that automate the design of a circuit must be interpretable and easy to use such that human efforts to apply the tools and algorithms are minimized. The black-box models and complex decision processes utilized by existing methods are hardly interpretable [5][7]. Beyond the generation of design solutions, information such as performance tradeoffs, design space partitioning information, and importance rankings (sensitivity analysis) of design variables provide utility.

To address the limitations of existing techniques, a novel batch-mode online optimization framework is developed to design analog integrated circuits through classification with adaptive labeling thresholds (*CALT*). The primary contributions of the work include: 1) the application of classifiers for both the modeling of the performance space and the sizing of an analog circuit, 2) the use of interpretable tree-based algorithms for surrogate modeling, and 3) a strategy to adaptively set the labeling thresholds for the training of the classifiers such that the lack of positively labeled data is resolved.

The rest of the paper is organized as follows. In Section II, methods utilized in *CALT* are analyzed. The framework and simulation results from the characterization of a low noise amplifier designed by executing *CALT* are presented in Section III. A discussion of the critical outcomes from analysis of the results is provided in Section IV. Some concluding remarks are provided in Section V.

2 PROPOSED METHODOLOGY

With *CALT*, the sizing of the components of an analog circuit is performed by the sequential completion of two tasks: 1) *multi-output classification* for performance modeling of a circuit, and 2) *optimization* for the generation of the component sizes for the circuit. The details of the classification framework that includes the adaptive labeling threshold strategy are provided in Section II-A. An analysis of the benefits of tree ensemble algorithms is provided in Section II-B, where random forest algorithms are adopted for surrogate predictive modeling. The framework to apply *design in the loop* is described in Section II-C. A summary of the *CALT* design flow is provided in Section II-D.

2.1 Classification with Adaptive Labeling Thresholds

Given the problem of sizing the components of an analog circuit, denote the design space as $X \subseteq \mathbb{R}^d$, and the performance space as $Y \subseteq \mathbb{R}^k$. Initially, a dataset $U = \{(x_1, y(x_1)), \ldots, (x_n, y(x_n))\} \in (X \times Y)^n$ is sampled from the design space. Latin Hypercube Sampling (*LHS*) [10] is applied, where *LHS* is a Monte Carlo method that provides a quasi-random sampling distribution. For a pre-specified sample size n, the design space is partitioned into equal regions, and a single point is randomly selected in each region.

After the initial dataset is generated, binary labels are assigned to each data point for each circuit performance metric based on whether a target threshold is met. The labeled space is denoted as $\hat{Y} \subseteq \{\pm 1\}^k$. The objective then becomes to train a classifier $h_k : X \rightarrow \hat{Y}_k$ for the k^{th} circuit performance metric that, given a new instance $x \in X$, predicts $\hat{y} = h_k(x) \in \hat{Y}_k$. A multi-output classification problem is, therefore, formulated.

A possible choice for the labeling threshold is the design specification. However, if the dimensionality of the design space is large, the initial dataset is unlikely to contain sufficient data points with positive labels for training. Instead, for a target specification, the labeling threshold is set to the ϵ^{th} percentile of the distribution of a given circuit performance metric in the dataset U as a lower bound, and the $(100-\epsilon)^{th}$ percentile of the distribution as the upper bound. If the corresponding specification exceeds the percentile value, the dataset contains enough positively labeled data points and the threshold is, therefore, set to the specification. Given the design specification set $S \subseteq \mathbb{R}^k$ for $s \in S$, the labeling threshold set $T \subseteq \mathbb{R}^k$ for $t \in T$ is generated as given by Algorithm 1. Binary labels are then assigned to each circuit performance metric based on whether the target set T is met, as given by Algorithm 2.

Algorithm 1: Adaptively Set the Labeling Thresholds

```
for i = 1 to k do
    if s_i is a lower bound for the i^th circuit performance
     metric y_i then
        t_i ← ε^th percentile of y_i in U;
        if t_i > s_i then t_i ← s_i;
    else
        t_i ← (100 − ε)^th percentile of y_i in U;
        if t_i < s_i then t_i ← s_i;
end end
```

Algorithm 2: Assign Labels for Classifier Training

```
for i = 1 to n do
    for j = 1 to k do
        if s_j is a lower bound for the j^th circuit performance
         metric y_j then
            if y_j(i) ≥ t_j then
                ŷ_j(i) = +1;
            else
                ŷ_j(i) = −1;
            end
        else
            if y_j(i) ≤ t_j then
                ŷ_j(i) = +1;
            else
                ŷ_j(i) = −1;
            end
        end
    end
end
```

Precision and *Recall* are utilized to evaluate the performance of the classifiers, which are defined as

$$Precision = \frac{Number\ of\ true\ positives}{Number\ of\ positive\ predictions}, \text{ and} \quad (1)$$

$$Recall = \frac{Number\ of\ true\ positives}{Number\ of\ positive\ instances}. \quad (2)$$

Combining *Precision* and *Recall* results in the *F1-score*, which is utilized as a single metric that evaluates the performance of a classifier, as given by (3).

$$F1\text{-}score = 2 \times \frac{precision \times recall}{precision + recall} \tag{3}$$

2.2 Applying Random Forest for Classification

In [11], decision tree (*DT*) algorithms [12] are applied to map from the circuit specifications to the circuit topology by utilizing past designs as reference. In this work, *DT*-based algorithms are utilized due to the following advantages:

- Tree-based models are fast to train while providing comparable prediction accuracy to other methods including neural networks,
- A small number of hyper-parameters require tuning, while data pre-processing is not necessary,
- Design space partitioning information is provided through the tree-structured models, and
- Feature importance rankings are generated.

To train a decision tree [12], the Gini index G_l is applied as the node splitting criteria, which is defined as

$$G_l = 1 - \sum_i f(i)^2, \tag{4}$$

where $f(i)$ is the fraction of positive instances for the i^{th} node split. The tree is grown by finding the largest reduction in the Gini index.

Ensemble techniques are applied to reduce model overfitting, which results from using single tree models. In this work, the random forest algorithm [13] is utilized, which draws samples with replacement from the dataset for the training of a bag of deep trees with a subset of the features. The final prediction is obtained by averaging the individual predictions produced by the models, as given by Algorithm 3.

Algorithm 3: Random Forest Algorithm

Let M = number of bootstrap samples ;
for *i=1 to M* **do**
  Create a bootstrap sample G_i of size N;
  Train a single tree on G_i with a randomly selected subset of features;
end
$\hat{y}(x) = \frac{1}{M} \times \sum_{i=1}^{M} \hat{y}_i(x)$

The execution of the random forest algorithm provides the importance ranking of the design variables[14]. During each iteration of bootstrap training, a single tree model is trained from the bootstrap samples and tested with the remaining samples. The comparison of the samples results in an out-of-Bag (*OOB*) error. The average of the *OOB* errors from all runs of bootstrap training is an estimate of the performance of the ensemble. Through random permutations of a feature set, the importance of a design parameter is determined by characterizing the impact of the changes on the *OOB* error, as described by the pseudocode provided as Algorithm 4.

2.3 Optimization-based Active Querying

After the classifiers for each performance metric are trained, qualified designs are determined from the intersection of the feasible regions of all models. A multi-objective search is executed for each iteration of the simulation loop to search for points such that the predicted probability scores of all models are simultaneously maximized. The candidate solutions are given as

$$x^* \in arg\max(p_1(x), ..., p_k(x)), \tag{5}$$

where $p_k(x)$ is the probability score predicted by the k^{th} classifier. The design points are then verified through SPICE simulation (*Cadence Virtuoso* in this work).

Algorithm 4: Feature Importance by Permutation

Let M = number of bootstrap samples;
for *each predictor variable j* **do**
  for *tree t, t=1 to M* **do**
    Get OOB error θ_t;
    Random permute observations of *j*;
    Get OOB error of the permuted set θ_j;
    $\theta_{jt} = \theta_j - \theta_t$;
  end
  Let $\mu(\theta_{jt})$ be the mean of θ_{jt} across all trees, and $\sigma(\theta_{jt})$ be the standard deviation of θ_{jt} across all trees;
  Feature importance of $j = \mu(\theta_{jt}) / \sigma(\theta_{jt})$;
end

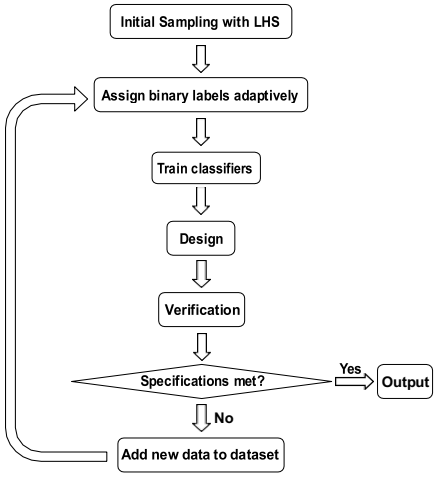


Figure 1: Proposed flow that applies classification with adaptive labeling thresholds to the design of an analog circuit.

2.4 Summary of the Design Flow of *CALT*

As shown in Fig. 1, the design flow of *CALT* includes six primary steps, which are described as

1. Initialization through the generation of random points in the design space with *LHS*. Execution of an automation script (*OCEAN*) to evaluate the performance of the circuit for each selected point with *SPICE* simulations,
2. Adaptively assigning a binary label to each performance metric of each selected point with Algorithm 1,
3. Training a random forest classifier for each performance metric with the dataset,
4. Running the multi-objective search algorithm *NSGA-II* on all of the model functions to generate design points and writing the resulting points to a data file,

5. Automation of the reading of the design points and execution of the SPICE simulations to evaluate the performance of the circuit with the generated component sizes, and

6. If no design point meets all of the specifications, add the verified data points to the dataset, and return to step 2.

3 SIMULATION RESULTS

CALT is applied to the design of an inductively degenerated differential low noise amplifier (LNA), which is shown in Fig. 2. The target operating frequency of the LNA is 2.4 GHz in a 65 nm technology.

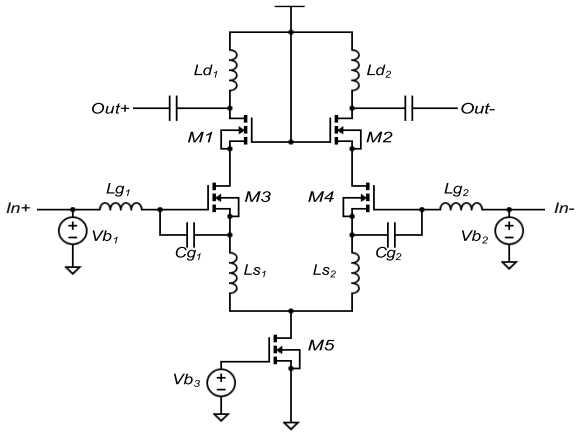


Figure 2: Circuit schematic of a differential low noise amplifier.

The design set consists of nine variables: the sizes of the inductors L_{g1}, L_{d1}, and L_{s1}, the widths of transistors M_1, M_3, and M_5, the size of capacitor C_{g1}, and the biasing voltages V_{b1} and V_{b3}. Due to the symmetry of the differential structure, the remaining variables are set to the same values as the corresponding counterparts. The transistor length is set to the minimum of 65 nm. The performance set includes the *power gain*, *noise figure* (*NF*), *third-order intercept point* (*IP3*), and *power consumption*. The target design variables are constrained as

$$\begin{aligned} &60\ nm \le transistor\ widths \le 900\ \mu m, \\ &\le \qquad\qquad \le \\ &0.01\ nH \quad inductor\ sizes \quad 12\ nH, \\ &30\ fF \le capacitor\ sizes \le 20\ pF, \text{ and} \\ &0\ V \le biasing\ voltages \le 1.2\ V. \end{aligned} \tag{6}$$

Two different sets of design specifications are targeted, the first given as Specification Set 1:

$$\begin{aligned} &Gain \ge 10\ dB, \\ &\le \\ &NF \quad 3\ dB, \\ &IP3 \ge -5\ dBm, \text{ and} \\ &Power \le 10\ mW, \end{aligned} \tag{7}$$

and the second as Specification Set 2:

$$\begin{aligned} &Gain \ge 14\ dB, \\ &NF \le 2.8\ dB, \\ &IP3 \ge -5\ dBm, \text{ and} \\ &Power \le 20\ mW. \end{aligned} \tag{8}$$

An initial dataset of 1000 points is sampled with *LHS*. After verifying that the dataset contains no points that satisfy all of the specifications, *CALT* is executed to solve for the nine design variables for both sets of target specifications, where ϵ, as described in Section 2.1, is set to 95. Five runs of *CALT* are executed for each of the target specification sets. A summary of the results is provided in Table I.

Table 1: Summary of Results for the design of the LNA with *CALT*

Parameters	Specification Set 1	Specification Set 2
M_1=M_2	363.1 µm	166.4 µm
M_3=M_4	165.9 µm	247.8 µm
M_5	109.4 µm	329.9 µm
L_{g1}=L_{g2}	8.39 nH	9.61 nH
L_{d1}=L_{d2}	5.23 nH	4.10 nH
L_{s1}=L_{s2}	1.03 nH	0.836 nH
C_{g1}=C_{g2}	0.468 pF	0.375 pF
V_{b1} = V_{b2}	0.661 V	0.633 V
V_{b3}	0.883 V	0.892 V
gain	10.75 dB	14.02 dB
NF	2.88 dB	2.79 dB
IP3	-4.75dBm	-4.63 dBm
power	8.96 mW	15.79 mW
min (max) Num. of iterations	3 (5)	9 (30)
avg Num. of iterations	4	17
min (max) Num. of samples	1217 (1357)	1630 (3100)
avg Num. of samples	1287	2190
min (max) execution time	4.2 hr (6.5 hr)	12.3 hr (41.1 hr)
avg execution time	5.4 hr	23.2 hr

As indicated by the results listed in Table I, qualified solutions are returned by *CALT* for both sets of target specifications. The Pareto fronts from the verified design points are provided in Fig. 3, where the tradeoff between the *gain* and *NF* of the LNA is shown. Since the *power budget* for Specification Set 2 is set to be 10 *mW* greater than that for Specification Set 1, the Pareto front for Specification Set 2 is closer to the upper-left corner of Fig. 3. In addition, an average of 17 iterations are needed to determine the design variables for Specification Set 2, as compared to four iterations required for Specification Set 1, as listed in Table I. Both specification sets are distinct since one targets lower *power consumption*, while the other targets a lower *NF* and higher *gain* by allowing for a higher *power budget*.

The total execution time for *CALT* consists of the time for initial sampling, offline model training and optimization, and verification through online simulation. As listed in Table I, the execution time of *CALT* is in the range of 4.2 to 6.5 hours for Specification Set 1, and 12.3 to 41.1 hours for Specification Set 2, which indicates a large variation in the convergence speed of the stochastic *CALT* design framework, especially for the more stringent parameter requirements of Specification Set 2.

The data plotted in Figs. 4 and 5 is from the run of the *CALT* algorithm with the fastest execution time. The changes in the labeling thresholds of the *gain* and *NF* for each executed iteration of the algorithm are shown in Fig. 4. The 95^{th} percentile of the *gain* distribution exceeds 10 dB after completion of the second iteration of the *CALT* algorithm when solving for Specification Set 1, and exceeds 14 dB after completion of the fourth iteration of the *CALT* algorithm for Specification Set 2. Thereafter, the labeling thresholds are maintained at 10 dB and 14 dB, respectively. In comparison, the 95^{th} percentile of the *IP3* distribution and the 5^{th} percentile of the *power* distribution exceed the corresponding target specifications beginning with the initial dataset, which indicates that the dataset

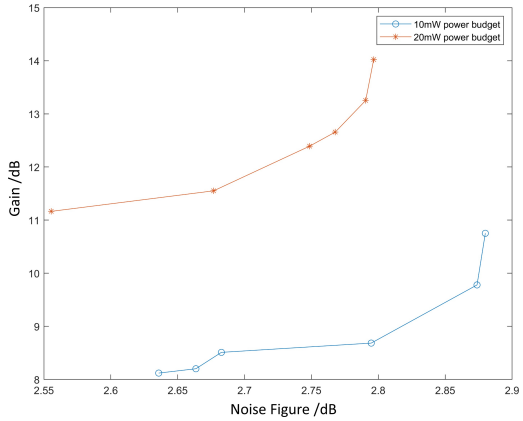


Figure 3: Generated pareto front between *gain* and *NF* when *power* and *IP3* constraints remain satisfied.

contains sufficient points with positive labels for the two circuit metrics. Therefore, the labeling thresholds for *IP3* and *power* are set to the corresponding specifications from the start of execution of the *CALT* design flow.

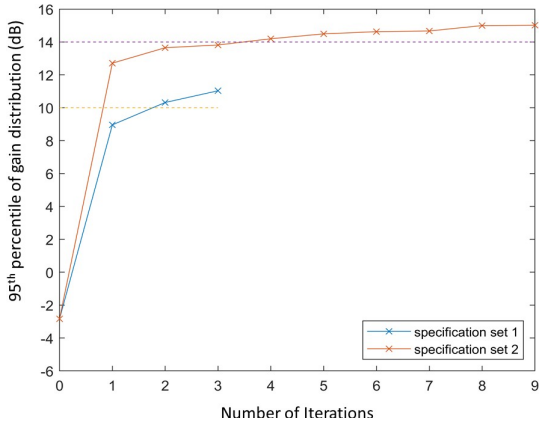
The *F1-scores* of the classifiers for each of the four performance metrics when solving for Specification Set 2 are shown in Fig. 5. The performance of the *gain* and *NF* classifiers is poor initially and improves as the number of iterations increases. In contrast, the performance of the *power* and *IP3* classifiers is relatively constant for all iterations. The difference in the performance of the classifiers is due to the change in the labeling thresholds when training the models for *gain* and *NF*. The results indicate that the convergence to qualified design solutions is shown to be correlated with the performance of the surrogate prediction models.

After sizing the components of the LNA with *CALT*, importance rankings of the design variables are extracted from the random forest models, as shown in Fig. 6. The size of inductor L_{s1} (L_{s2}) results in the greatest impact on all metrics except for the *power consumption*, as L_s is critical for input matching. The rankings also reveal the significant impact of the two biasing voltages V_{b1} (V_{b2}) and V_{b3} on *IP3* and *power consumption*, which are large-signal circuit performance metrics. The automatically extracted importance rankings allow for the narrowing of the input search space to a small set of critical design variables. In comparison, for manual custom design, both analytic formulae and design expertise are needed to identify the critical design variables best suited for the optimization of a performance metric.

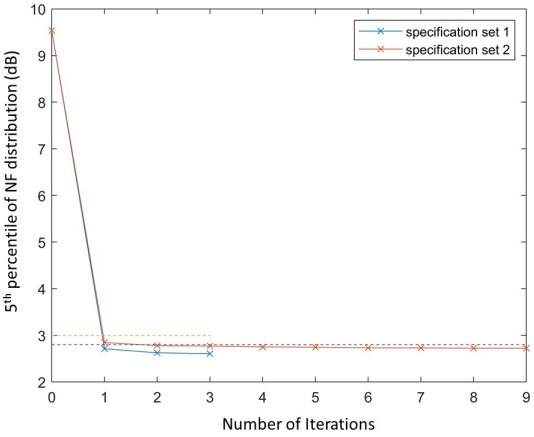
As a final step, decision trees are trained with the final dataset. A tree for *NF* prediction trained with the final dataset generated from completion of the CALT sizing methodology on Specification Set 2 is shown in Fig. 7. The design space is partitioned by the tree model, and decision paths are shown that serve as criteria on whether a design point is expected to satisfy the specified *NF*.

4 DISCUSSION

If the topology and technology node are fixed, the design space of an analog circuit is also fixed. The necessary partitioning details of the design space are, therefore, learned by *CALT* from simulation data. With the binary classifiers, decision boundaries between feasible and infeasible regions are identified for a given specification. The optimizations are used to search for design points in the common feasible regions of all models. As new design points



(a)



(b)

Figure 4: Change in a) the 95^{th} percentile of the *gain* distribution and b) the 5^{th} percentile of the *NF* distribution with each iteration of the *CALT* algorithm. The results are used to determine the labeling threshold of the *gain* and *NF*.

are actively queried, more information on both the design space and the performance space is gathered. The performance of the classifiers, therefore, improves, which results in the convergence to a design solution.

Fine-tuning of the surrogate models is performed with the proposed closed-loop learning system. The dataset determined during the final iteration of the sizing flow is considered as the minimum required for convergence to a design solution. The *CALT* framework is driven by the circuit specifications, which allows for customized designs of analog circuits, where the specifications are adjusted based on the design needs.

5 CONCLUSIONS

A simulation-based framework for classification with adaptive labeling thresholds (*CALT*) is proposed to automatically size the components of an analog circuit. Classifiers are applied to check whether the target specifications are satisfied. The binary labeling

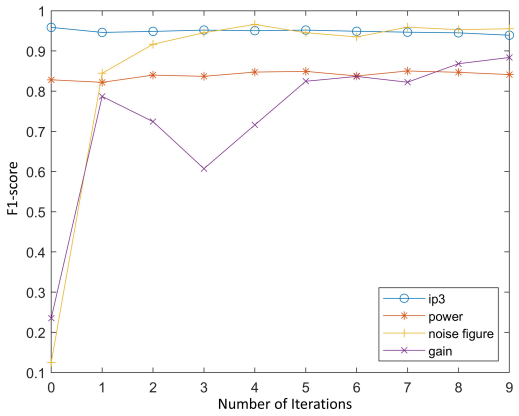


Figure 5: Cross-validated F1-scores of the random forest classifiers for the *gain*, *NF*, *IP3*, and *power*, when targeting Specification Set 2.

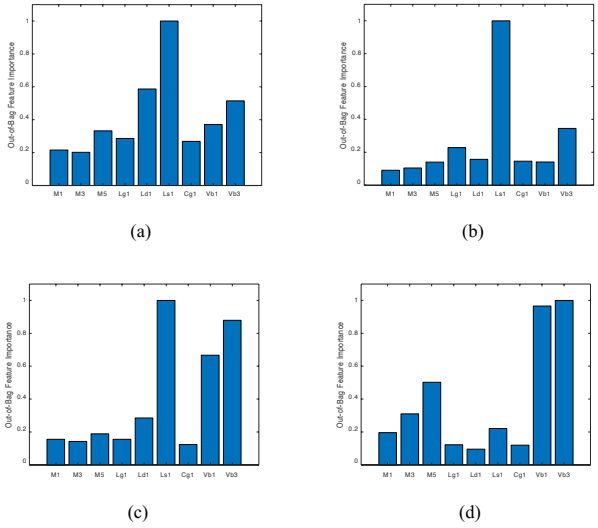


Figure 6: Variable importance rankings extracted for a) *gain*, b) *NF*, c) *IP3*, and d) *power* for target Specification Set 2 after completion of the CALT circuit sizing flow.

thresholds are adaptively adjusted based on a target percentile of a circuit performance metric characterized by the dataset. Random forest classifiers are executed for surrogate modeling that offer a feature importance ranking of the design variables. *CALT* is applied to the design of an LNA for two sets of target specifications. Qualified design solutions are generated for two sets of specifications with an average execution of 4 and 17 iterations of the optimization loop, which require an average of 1287 and 2190 simulation samples, and an average execution time of 5.4 hours and 23.2 hours, respectively. *CALT* is a specification-guided analog sizing flow that offers interpretable models and achieves high sample efficiency without requiring the use of prior circuit models.

ACKNOWLEDGMENTS

This work was supported in part by the Air Force Research Laboratory under Contract FA8075-14-D0025/DSTAT-15-1196, and in part by the National Science Foundation under Grant CNS-1751032.

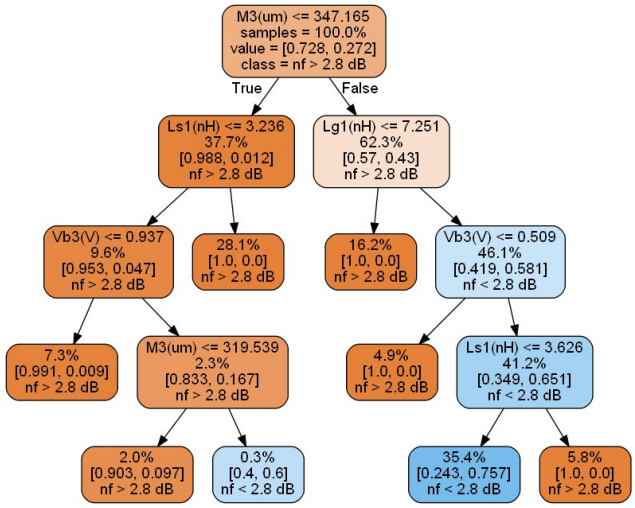


Figure 7: A decision tree for *NF* prediction trained with the final dataset after the completion of the component sizing flow targeting Specification Set 2.

REFERENCES

[1] J. Crossley, A. Puggelli, Hanh-Phuc Le, B. Yang, R. Nancollas, K. Jung, L. Kong, N. Narevsky, Y. Lu, N. Sutardja, E. J. An, A. Vincentelli, and E. Alon, "BAG: A designer-oriented integrated framework for the development of AMS circuit generators," *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pp. 74–81, Nov. 2013.

[2] M. Barros, J. Guilherme, and N. Horta, "Analog circuits optimization based on evolutionary computation techniques," *Integration, the VLSI Journal*, Vol. 43, No. 1, pp. 136–155, Jan. 2010.

[3] R. Phelps, M. Krasnicki, R. A. Rutenbar, R. Carley, and J. R. Hellums, "Anaconda: Simulation-based synthesis of analog circuits via stochastic pattern search," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 19, No. 6, pp. 703–717, Jul. 2000.

[4] W. Lyu, F. Yang, C. Yan, D. Zhou, and X. Zeng, "Multi-objective bayesian optimization for analog/RF circuit synthesis," *Proceedings of the IEEE/ACM Design Automation Conference*, pp. 1–6, Jun. 2018.

[5] K. Hakhamaneshi, N. Werblun, P. Abbeel, and V. Stojanovic, "BagNet: Berkeley analog generator with layout optimizer boosted with deep neural networks," *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pp. 1–8, Jul. 2019.

[6] R. Vemuri and G. Wolfe, "Adaptive sampling and modeling of analog circuit performance parameters with pseudo-cubic splines," *Proceedings of the IEEE/ACM International Conference on Computer Aided Design*, pp. 931– 938, Dec. 2004.

[7] Z. Gao, J. Tao, F. Yang, Y. Su, D. Zhou, and X. Zeng, "Efficient performance trade-off modeling for analog circuit based on Bayesian neural network," *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pp. 1–8, Nov. 2019.

[8] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, Vol. 6, No. 2, pp. 182–197, Jan. 2002.

[9] F. Bernardinis, M. Jordan, and A. Sangiovanni-Vincentelli, "Support vector machines for analog circuit performance representation," *Proceedings of the IEEE/ACM Design Automation Conference*, pp. 964–969, Jan. 2003.

[10] M. Mckay, R. Beckman, and W. Conover, "A comparison of three methods for selecting values of input variables in the analysis of output from a computer code," *Technometrics*, Vol. 21, No. 2, pp. 239–245, May 1979.

[11] T. Mcconaghy, P. Palmers, G. Gielen, and M. Steyaert, "Automated extraction of expert knowledge in analog topology selection and sizing," *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pp. 392–395, Nov. 2008.

[12] W. Loh, "Classification and regression trees," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, Vol. 1, No. 1, pp. 14–23, Jan. 2011.

[13] T. Ho, "A data complexity analysis of comparative advantages of decision forest constructors," *Transactions on Pattern Analysis and Applications, Springer*, Vol. 5, No. 2, pp. 102–112, Jun. 2002.

[14] L. Breiman, "Random Forests," *Machine Learning*, Vol. 45, No. 1, pp. 5–32, Oct. 2001.