Using Artificial Neural Networks to Size Analog Integrated Circuits

João Rosa
Instituto de Telecomunicações
Instituto Superior Técnico
Lisboa, Portugal

joao.silva.rosa@tecnico.ulisboa.pt

***Abstract*— The work presented in this dissertation belongs to the scientific area of electronic design automation and addresses the automatic sizing of analog integrated circuits. Particularly, this work explores an innovative approach to automatic circuit sizing using deep learning and artificial neural networks to learn patterns from previously optimized design solutions. In opposition to classical optimization-based sizing strategies, where computational intelligent techniques are used to iterate over the map from devices' sizes to circuits' performances provided by design equations or circuit simulations, artificial neural networks are shown to be capable of solving analog integrated circuit sizing as a direct map from specifications to the devices' sizes. Two separate artificial neural network architectures are proposed: a Regression-only model and a Classification and Regression model. The goal of the Regression-only model is to learn design patterns from the studied circuits, using circuit's performances as input features and devices' sizes as target outputs. This model can size a circuit given its specifications for a single topology. The Classification and Regression model has the same capabilities of the previous model, but it can also select the most appropriate circuit topology and its respective sizing given the target specification. The proposed methodology was implemented and tested on two analog circuit topologies. The achieved results show that the trained artificial neural networks were able to extend the circuit performance boundaries outside the train/ validation set, showing that, more than a mapping from the training data, the model is actually capable of learning reusable design patterns.**

Keywords*—*Analog Integrated Circuit Design, Electronic Design Automation, Deep Learning, Artificial Neural Networks

I. Introduction

While integrated circuits (IC) are mostly implemented using digital circuitry, analog and radio-frequency (RF) circuits are still necessary and irreplaceable in the implementation of most interfaces and transceivers. However, unlike the digital design where an automated flow is established for most design stages, the absence of effective and established computer-aided-design (CAD) tools for electronic design automation (EDA) of analog and RF IC blocks poses the largest contribution to their bulky development cycles, leading to long, iterative and error-prone designer's intervention along the entire course of the design flow [1].

The prevalent method to analyse and evaluate analog ICs is focused on the mapping from the devices' sizes to the circuit's performance figures. This mapping can be done using approximate equations, which usually support manual design approaches, or, using the circuit simulator, that is used to verify and fine tune the manual design. For automatic sizing, simulation-based optimization is the most prevalent method in both industrial [2][3] and academic [4][5] environments.

This work is funded by FCT/MEC through national funds and when applicable co-funded by FEDER – PT2020 partnership agreement under the project UID/EEA/50008/2013, and by the Fundação para a Ciência e Tecnologia (Grants SFRH/BPD/104648/2014, SFRH/BPD/ 120009/2016 and SFRH/BD/133662/2017).

Hence, instead of going from the target specification to the corresponding device sizes, the designer or the EDA tool are actually evaluating the inverse of the problem countless times, i.e., trying combinations of design variables to find a sizing such that the circuit meets specifications, as shown in Fig.1 (a). In this work we try to address this issue, performing an exploratory research on how artificial neural networks (ANNs) and deep learning [6] can solve the circuit sizing problems directly, as shown in Fig. 1 (b), given, of course, the appropriate training set.

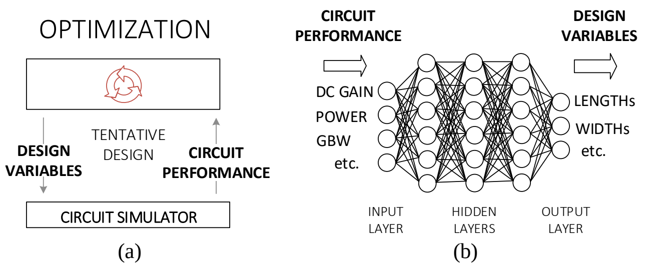

Fig. 1. Illustration of how diferent methods address the analog IC sizing problem, and, find the dimentions of the devices so that the circuit meet the specifications. (a) Optimization-based sizing: Inverse approach; (b) ANNs: Direct aproach.

ANNs were already used in some EDA methodologies for analog IC sizing [7][8], but to replace/ complement the circuit simulator in the optimization-based sizing approach of Fig. 1 (a). While the evaluation speed is greatly increased by avoiding time-consuming circuit simulations, the accuracy lost is only recovered by returning to the circuit simulator in later stages of the optimization. Moreover, training these models is done over the entire design space, which spends valuable resources modelling and evaluation large regions of unusable design combinations. In [9] this aspect is somewhat addressed, but still, the ANNs were trained to replace the simulator, instead of being trained to size the circuit for a given specification. In [10], a prediction method of element values for required specifications using deep learning was proposed. The goal of this model was to predict element values that realize desired circuit characteristics, which is similar to the model presented in this paper.

In this work, we explore how an ANN that is trained using circuit sizing solutions from previous optimizations can learn the design patterns of the circuit, and, as shown in Section IV, can even generate sizing for efficient circuits on specifications outside those in the training dataset.

The paper is organized as follows. In Section II, the design flow of analog IC sizing in the context of deep learning, is presented. In Section III, analog IC sizing is formulated as a machine learning problem and the dataset is defined. In Section IV, the Regression-only Model, including data pre-processing, normalization, model and hyper-parameter tuning, training, and, finally, a method to sample analog IC sizing from the trained ANNs, are described. In Section V, the proposed method is used in the sizing of an analog amplifier. In Section VI, the Classification and Regression Model, as well as hyper-parameter tuning training of the ANNs, are described. In Section VII, the results obtained with the model are presented. Finally, Section VIII concludes the article.

II. Design Flow

The proposed automatic design flow of analog IC sizing using ANNs is as follows:

1. Determine the prediction target;
2. Collect data for learning;
3. Create learning model;
4. Train learning model;
5. Confirm the accuracy of the prediction;
6. Sample the obtained results;
7. Test the sampled results in AIDA.

The first step in the creation of an ANN model is to determine the prediction target. In this case, we want the network to learn design patterns from the studied circuits, using circuit's performances (DC Gain, current consumption (IDD), gain bandwidth (GBW) and phase margin (PM)) as input features and devices' sizes (such as such as widths and lengths of resistors and capacitors) as target outputs. The next step involves gathering data so that the model can learn patterns from the input-output mapping. Data should be split into three different sets: the training set, the validation set and the test set. After determining the prediction target and assembling the data, hyper-parameters of the model should be selected. These are the number of layers, number of nodes per layer, the activation functions, and the loss function, to name a few. After selecting the most appropriate hyper-parameters, the model is ready to be trained. At this stage, the model will attempt to iteratively find its optimal network weights. After training the model and achieving a sufficiently low error on the training phase and on the validation set, the model is ready to be used for predicting the output of real-world input data. This is where we will evaluate the accuracy of the predicted results and obtain devices' sizes from a set of desired circuit performances. The next step involves sampling the results from the model. This step is essential to circumvent possibly biased solutions predicted by the ANN. In the final step, we test the feasibility of the obtained solutions in AIDA.

AIDA, developed in the Integrated Circuits Group at Instituto Superior Técnico, is an analog integrated circuit design automation environment, which implements a design flow from a circuit-level specification to physical layout description..

III. Problem and Dataset Definition

Let $V_{<i>} \in \mathbb{R}^N$ be the vector of design variables that define the sizing of the circuit, where the index i inside the chevron identifies solution point i in the dataset, and $S_{<i>} \in \mathbb{R}^D$ be the vector of the corresponding circuit performance figures. The ANNs presented in this work were trained to predict the most likely sizing given the target specifications, as shown in (1).

$$V_{<i>} \sim \text{argmax}(\text{P}(V_{<i>}|S_{<i>})) \qquad (1)$$

Hence to train the ANN, the training data is comprised of a set T, of M data pairs $\{V, S\}_{<i>}$. Since we want the model to learn how to design circuits properly, these pairs must correspond to useful designs, e.g., optimal or quasi-optimal design solutions for a given sizing problem.

A. Inequality Specifications and Data Augmentaiton

While the previous definition allows to train a model suitable for analog IC sizing, circuits' target specifications are, more often than not, defined as inequalities instead of equalities. Therefore, an ANN trained to map $S \rightarrow V$ may have difficulties extrapolating to some specification values that are actually worse than the ones provided in training.

The point is, if the sizing $V_{<i>}$ corresponds to a circuit whose performance is $S_{<i>}$, then it is also a valid design for any specifications whose performance targets, $S'_{<i>}$, are worse than $S_{<i>}$. Having this in mind, an augmented dataset, T' can be obtained from T as the union of T and K copies of T, as indicated in (2), where the for each sample i, the $S_{<i>}$ is replaced by $S'_{<i>}$ according to (3).

$$T' = \{T \cup T^{C1} \cup T^{C2} \cup T^{CK}\} \qquad (2)$$

$$S'_{<i>} = S_{<i>} + \left(\frac{\gamma}{M} \sum_{j=1}^{M} S_{<j>} \right) \Delta\Gamma \qquad (3)$$

Where, $\gamma \in]0, 1[$ is a factor used to scale the average performances, Δ is a diagonal matrix of random numbers between [0, 1], and, $\Gamma \in \{-1, 1\}^D$ is the target diagonal matrix that define the scope of usefulness of the circuit. Its diagonal components take the value -1 for performance figures in which a smaller target value for the specification is also fulfilled by the true performance of the design, e.g., DC Gain, and, the value 1 is for the diagonal components corresponding to performance figures that meet specification targets that are larger than the true performance of the circuit, e.g., power consumption.

IV. Regression-Only Models for Analog IC Sizing

The ANNs models considered in this work consider fully connected layers without weight sharing. Given the number of features that are used in the model and the size of the datasets that will be considered in this application, the model is not very deep, containing only a few of hidden layers. A base structure for this model can be observed in Fig. 2. The best structure depends of the dataset, but a systematic method is proposed later in this Section to specify such models. To train and evaluate the model, the datasets are split in training (80%-90%) and validation sets (20%-10%). An additional small test set, whose specifications are drawn independently, are used to verify the real-world application of the model.

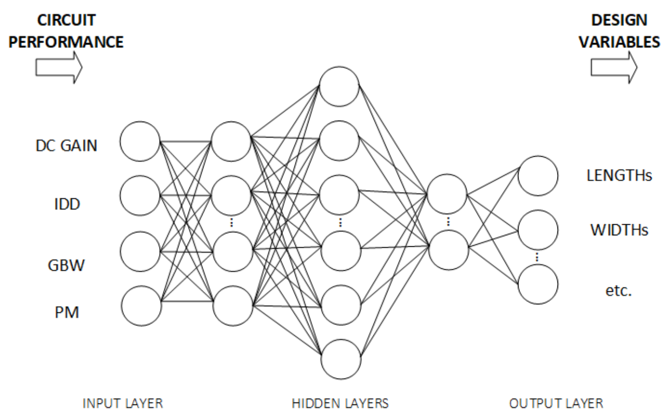

Fig. 2. Base structure of the Regression-only Model.

A. *Polynomial Features and Data Normalization*

To increase training efficiency, the ANN is trained with an input X that is the feature mapping Φ of S normalized. Each input data sample $X_{<i>}$ is given by:

$$X_{<i>} = \frac{\Phi(S_{<i>}) - \mu_\Phi}{\sigma_\Phi} \tag{4}$$

where $\Phi(S_{<i>})$ is a second order polynomial mapping of the original specifications, e.g., for $S_{<i>}$ = [a, b, c], $\Phi(S_{<i>})$ = [a, b, c, a^2, a*b, a*c, b^2, b*c, c^2]; μ_Φ is the mean of the components of Φ, and, σ_Φ is the standard deviation of the components of Φ. The output of the network, Y, is defined from V by (5):

$$Y_{<i>} = \frac{V_{<i>} - \min(V)}{\max(V) - \min(V)} \tag{5}$$

B. *Model Structure and Hyper-parameter tuning*

The guidelines that were used to select the hyper-parameters for the ANN Regression-only Model architecture were as follow:

- The number of input nodes is 15 (obtained from the second order polynomial feature extension of 4 performance figures). After applying gridsearch (i.e. iterative method that exhaustively considers all parameter combinations that the designer wishes to explore), the number of selected hidden layers was 3. As a rule of thumb, the number of nodes should increase in the first hidden layer to create a rich encoding (120 nodes), and then, decreases toward the output layer to decode the predicted circuit sizing (240 and 60 nodes in the second and third hidden layers, respectively). The number of output nodes depends on the number of devices' sizes from the topology being studied (e.g. 12 for VCOTA and 15 for Two Stage Miller);
- In initial testing, Sigmoid was used as the activation function of all nodes from all layers, except the output layer, but ReLU ended up being the preferred choice. The gradient using ReLU is better propagated while minimizing the cost function of the network. Output layer nodes don't use activation functions because we wish to find the true values predicted by the network instead of approximating them to either end of an activation function;
- Initially, the SGD optimizer was used, but later dropped in favour of Adam, which has better documented results;
- The models were first designed to have good performance (low error) in the training data, even if overfitting was observed. This method allowed to determine the minimum complexity that can model the training data. Overfitting was then addressed using L2 weight regularization;
- Initial random weights of the network layers were initialized by means of a normal distribution;
- Model performance was measured through a Mean Absolute Error (MAE) loss function, while the training of the model was performed using a Mean Squared Error (MSE) loss function;
- A high number of epochs (5000) was chosen for the training of early networks to ensure that the training reached the lowest possible error. One epoch occurs when all training examples execute a forward pass and a backward pass through the network. It is often a good practice to train the network for a high number of epochs, save the network weights from the training, and perform future testing using those weights with a lower number of epochs (e.g. 500);
- A variable number was chosen for batch size, between 256 and 512, depending on the number of epochs. Batch size is the number of training examples in one forward pass/ backward pass;
- Finally, *gridsearch* is done over some hyper-parameters (number of layer, number of nodes per layer, non-ideality and regularization factor) to fine tune the model.

C. *Training*

The loss function, L, of the model that is optimized during training is the MSE of the predicted outputs Y' with respect to the true Y plus the $L2$ norm of the model's weights, W, times the regularization factor λ, according to (6).

$$L = \frac{1}{M}\sum_{j=1}^{M}\left(\left(Y'_{<j>} - Y_{<j>}\right)^T\left(Y'_{<j>} - Y_{<j>}\right)\right) + \lambda\|W\|^2 \tag{6}$$

The training of the models is done using Adam [12], a variant of stochastic steepest descent with both adaptive learning rate and momentum that provides good performance, moreover, it is quite robust with respect to its hyper-parameters. Other error metrics such as MAE are also considered when validating the results.

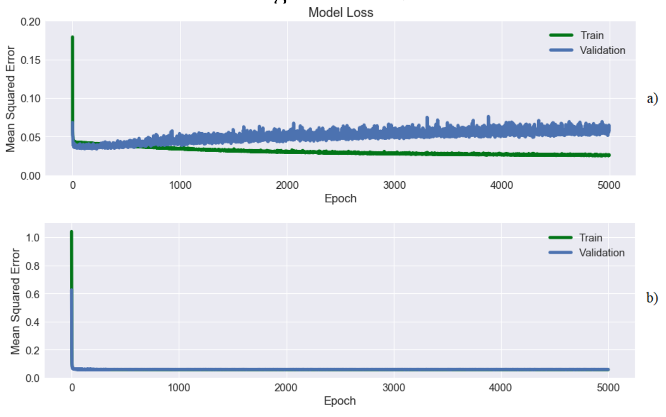

Fig. 3. Evolution of prediction error on train and validation sets during training: (a) ANN that overfits the training data, showing high error on the validation set. (b) Same ANN trained with *L2* norm weigh regularization, showing better performance on the validation set.

Transfer learning is a common practice in deep learning community, where an ANN trained with a large dataset are repurposed for other, similar, applications where data is scarce or, training is more expensive [6]. For EDA, transfer learning create opportunities in technology and/or topology migration, where an ANN trained for a unctional block might be able to accelerate and reduce the data required to extend their applicability to other technology nodes, corners and variability awareness, and/or other circuit topologies.

D. Sampling from the ANN

Sampling from the ANN is done using (3) *P* times, with *Γ* replaced by *–Γ*, i.e., we ask the model to predict a set of *P* sizing solutions given circuit performances that are better than the desired specifications. If not all performance figures used to train the model are specified, then the corresponding component in the diagonal random matrix Δ from (3) should be a random value in the range of [-1, 1]. For instance, if the target specifications are gain bandwidth product (GBW) over 30 MHz and current consumption (IDD) under 300 µA, and, the model was trained with DC Gain, GBW and IDD. A set of circuit performances given to the ANN could be, e.g., {(50dB, 35MHz, 290µA), (75dB, 30MHz, 285µA), (60dB, 37MHz, 250µA), …, (90dB, 39MHz, 210 µA)}.

The reasoning behind this sampling is that even if the ANN has properly learned the designs patterns present in the performances of the sizing solutions in the training data, when the performance trade-off implied by the target specifications being requested are not from the same distribution than the training data, the prediction of the ANN can be strongly badly biased. While using the augmented dataset described in Section II.B alleviates this bias, it is still better to sample the ANN this way. The selection of solutions from the *P* predictions of the ANN is done by simulating the predicted circuit sizing, and, either using a single value metric, such as some Figure-of-Merit (FoM) for the circuit, select the most suitable solution, or, using some sort of Pareto dominance to present a set of solutions exploring the trade-off between specifications.

V. Case Study 1

For proof of concept, the amplifier using voltage combiners for gain enhancement (VCOTA) from [13] was used, and, for a second example, a two stage Miller amplifier was considered, The circuits' schematic are shown in Fig. 4, showing the devices with annotated design variables.

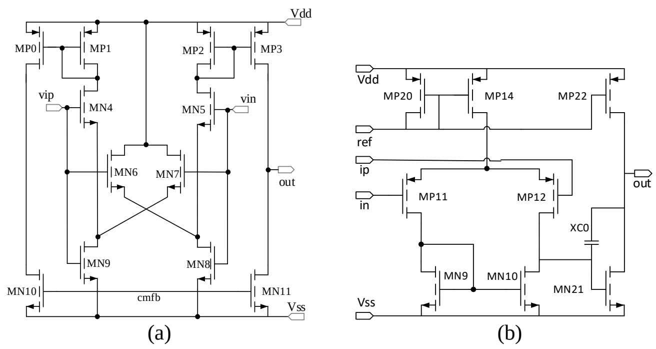

Fig. 4. Circuit schematic showing the devices and corresponding design variables: (a) Single stage amplifier with gain enhancement using voltage combiners; (b) Two Stage Miller amplifier.

A. Dataset

Datasets were obtained from a series of previously done studies on this circuit for the UMC 130 nm technology design process, and contain only optimized circuit sizing solutions.

For proof of concept, the first study was performed on the VCOTA topology. Dataset for this example, i.e. Dataset-1, before any augmentation has 16,600 different design points. The second study was performed on the Two Stage Miller topology. For this example, the dataset before any augmentation, i.e. Dataset-2, has 5162 different design points. The circuit performances that were considered to train the ANN in both cases were DC Gain, IDD, GBW and PM, and the ranges of values found in the dataset are shown in Table I.

TABLE I. Performance Ranges in the DataSet

		DC Gain	**GBW**	**IDD**	**PM**
VCOTA	***Max***	56.8 dB	78 MHz	395 uA	80
	Min	44.7 dB	34 MHz	221 uA	60
Two Stage Miller	**Max**	97.2 dB	102.8 MHz	0.8 uA	89.9º
	Min	59.8 dB	1.5 MHz	0.3 uA	55º

B. ANNs Structure and Training

Three ANNs were trained for the VCOTA topology, i.e., ANN-1 to ANN-3. The structure considered has 15 input variables (obtained from the second order polynomial feature extension of the 4 performance figures from Table I), 3 hidden layers with 120, 240, 60 nodes each, and, the output layer has 12 nodes.

All ANNs were implemented in Python with Keras, using TensorFlow [14-18] as backend. The code was run, on an Intel® Core™ i7 Quad CPU 2.6 GHz with 8 GB of RAM.

ANN-1, was trained on the original dataset, for 5000 epochs (passes through the entire dataset) with batches of 512 samples. Its training took less than 15 minutes. ANN-2, was trained on the dataset augmented 40 times (almost 700K samples) for the same 5000 epochs. Its training took approximately 8 hours. ANN-3 was also trained on the same augmented dataset, but only for 500 epochs, but was initialized with weights from ANN-1. Its training took less than an hour. Their performance after training on the training and validation sets is summarized in Table II.

TABLE II. Performance of Trained ANNs for the VCOTA topology

	MSE Train	**MSE Val.**	**MAE Train**	**MAE Val.**
ANN-1	0.0159	0.0157	0.0775	0.0776
ANN-2	0.0124	0.0123	0.0755	0.0750
ANN-3	0.0124	0.0124	0.0754	0.0753

Table III indicates the average MAE between all the predicted and true devices' sizes from the test set.

TABLE III. Average MAE between the predicted and true devices' sizes for the VCOTA topology

Devices' sizes items	**MAE**		
	ANN-1	**ANN-2**	**ANN-3**
_w8	2.75E-09	2.70E-09	2.65E-09
_w6	1.48E-05	1.62E-06	1.62E-06
_w4	5.24E-06	6.05E-06	6.04E-06
_w10	4.33E-06	4.59E-06	4.51E-06
_w1	8.91E-07	2.22E-06	2.17E-06
_w0	1.06E-05	1.84E-06	1.83E-06
_l8	2.98E-08	3.41E-08	3.35E-08
_l6	1.39E-07	1.56E-08	1.56E-08
_l4	1.21E-07	1.37E-08	1.36E-08
_l10	6.61E-08	6.99E-08	7.05E-08
_l1	5.32E-08	9.73E-08	9.98E-08
_l0	2.11E-08	5.08E-08	5.07E-08

In terms of performance, ANN-2 and ANN-3 showed the best results. From Table II, we can observe that MSE and MAE error for the training and validation sets were lower for these networks, when compared to ANN-1, but not by a long margin. In terms of individual MAE for each devices' sizes, results were very similar across all ANNs, with a slight advantage for ANN-1.

Three other ANNs were trained for the Two Stage Miller topology, i.e. ANN-4 to ANN-6. The structure considered has 15 input variables (obtained from the second order polynomial feature extension of the 4 performance figures from Table I**Erro! A origem da referência não foi encontrada.**), 3 hidden layers with 120, 240, 60 nodes each, and, the output layer has 15 nodes.

ANN-4, was trained on the original dataset, for 5000 epochs with batches of 512 samples, taking approximately 12 minutes to conclude the training. ANN-5 was trained on the dataset augmented 20 times (more than 100K samples) for 500 epochs. Its training took approximately 20 minutes. The training set was comprised of 96% of total data, while the validation set of 4%. ANN-6, was trained on the dataset augmented 20 times, for 500 epochs with batches of 512 samples, initialized with weights from ANN-1. The training set and the validation set were split with a 50% ratio (since the dataset was augmented, there is no problem in choosing the same percentage for both sets). Its training took approximately 20 minutes. Their performance after training on the training and validation sets is summarized in Table IV.

TABLE IV. PERFORMANCE OF TRAINED ANNS FOR THE TWO STAGE MILLER TOPOLOGY

	MSE Train	MSE Val.	MAE Train	MAE Val.
ANN-4	0.0561	0.0414	0.1357	0.0937
ANN-5	0.0072	0.0073	0.0590	0.0595
ANN-6	0.0072	0.0073	0.0590	0.0597

Table V indicates the average matching rate between all the predicted and true devices' sizes from the test set.

TABLE V. AVERAGE MATCHING RATE BETWEEN THE PREDICTED AND TRUE DEVICES' SIZES FOR THE TWO STAGE MILLER TOPOLOGY

Devices' sizes items	**MAE**		
	ANN-4	**ANN-5**	**ANN-6**
_wb	2.75E-06	1.27E-05	2.86E-06
_wp	8.59E-06	7.76E-05	8.85E-06
_wal	4.01E-06	2.78E-05	4.20E-06
_w2g	8.76E-06	7.35E-05	8.86E-06
_lb	7.77E-07	4.65E-06	8.15E-07
_lp	6.06E-07	4.54E-06	6.11E-07
_lal	9.41E-07	8.09E-06	9.47E-07
_l2g	6.57E-07	5.43E-06	6.67E-07
_mbp	5.64E-01	4.27E-00	5.89E-01
_mb2	6.49E-01	8.91E-00	6.55E-01
_mal	1.69E+01	9.34E+01	1.77E+01
_mp	2.84E+01	1.35E+02	2.86E+01
_m2g	2.84E+01	1.30E+02	2.84E+01
_lc	1.18E-05	7.49E-05	1.20E-05
_nfc	2.30E+01	1.44E+02	2.31E+01

In terms of performance, ANN-5 and ANN-6 showed the best results. From Table IV, we can observe that MSE and MAE for the training and validation sets were significantly lower for these networks, when compared to ANN-4. From Table V, we can see that ANN-5 obtained lower error on the first 8 items, but performed worse on the remaining items, which yielded better results on the other two networks.

C. Sampling the ANNs for new desings

Sampling the ANNs was done as described in Section III D, with P = 40 and γ = 0.15, i.e., 100 random samples with a deviation of up to 15% from the specifications. Selection of the best solution was done by FOM for target 1, GBW for target 2, and IDD[a] and FOM[b] for target 3. Is easily seen by the performance of the obtained circuits that the ANNs learned the design patterns and can even extrapolate for specifications outside those of the training data. Moreover, circuit with FoMs larger than 1000 were obtained in all samplings of the ANNs. FoM is used in this example, as it was an optimization target in the process used to obtain the dataset.

Observing Table VI, we conclude ANN-1 can explore easily new specifications, but generates greater variability and worse designs when sampled. ANN-2 and ANN-3 are more stable generating always good designs when sampled inside the training data. ANN-2 shows more limitations when trying to explore new specifications. ANN-3, because it used transfer learning from ANN-1, is more flexible to new specifications, but still lags when compared to ANN-1.

TABLE VI. PERFORMANCE OF SAMPLED DESIGNS FOR THE VCOTA TOPOLOGY

	#HF[a]	**DC Gain**	**GBW**	**IDD**	**PM**	**FOM [b]**
Target 1		50 dB	60 MHz	300 uA	65º	
ANN-1	0.33	50 dB	63 MHz	318 uA	64º	1180
ANN-2	1	51 dB	61 MHz	320 uA	65º	1153
ANN-3	1	51 dB	63 MHz	325 uA	65º	1165
Target 2		40 dB	150 MHz	700 uA	55º	
ANN-1	0.24	44 dB	148 MHz	822 uA	54º	1082
ANN-2	0.21	49 dB	60 MHz	325 uA	73º	1106
ANN-3	1	43 dB	100 MHz	509 uA	61º	1182
Target 3		50 dB	30 MHz	150 uA	65º	
ANN-1 [c]		50 dB	3 MHz	141 uA	74º	116
ANN-1	0.73	50 dB	30 MHz	205 uA	74º	889
ANN-1 [d]		49 dB	67 MHz	329 uA	60º	1215
ANN-2 [c]	1	54 dB	38 MHz	240 uA	71º	950
ANN-2 [d]		54 dB	46 MHz	268 uA	64º	1033
ANN--3 [c]	0.97	55 dB	30 MHz	217 uA	69º	842
ANN-3 [d]		54 dB	54 MHz	309 uA	56º	1050

[a] Ratio of the number of solutions with FOM higher than 850 (the min value in the training data was 900) to the total number of samples; [b] MHz.pF/mA; [c] Best IDD; [d] Best FOM.

Observing Table VII, we conclude that ANN-4 and ANN-6 are capable of generating stable designs, despite being inaccurate for some items. ANN-2 shows some limitations for one target, but can generate designs similar to the other networks for the remaining targets.

TABLE VII. PERFORMANCE OF SAMPLED DESIGNS FOR THE TWO STAGE MILLER TOPOLOGY

	DC Gain	GBW	IDD	PM
Target 1	70 dB	10 MHz	30 uA	65º
ANN-4	77 dB	9 MHz	39 uA	65º
ANN-5	78 dB	9 MHz	35 uA	64º
ANN-6	77 dB	8 MHz	36 uA	64º
Target 2 [a]	100 dB	10 MHz	30 uA	65º
ANN-4	95 dB	12 MHz	48 uA	89º
ANN-5	87 dB	6MHz	41 uA	68º
ANN-6	85 dB	13 MHz	39 uA	42º
Target 3 [a]	70 dB	2 MHz	10 uA	65º
ANN-4	80 dB	3 MHz	29 uA	70º
ANN-5	75 dB	1 MHz	19 uA	75º
ANN-6	72 dB	1 MHz	18 uA	75º

[a] Targets outside the performances present in the dataset.

VI. Classification and Regression Model For Analog IC Sizing

The ANN architecture considered in this Section is similar to the one used for the Regression-only Model, but now there is an increased number of output nodes, as seen in Fig. 5. The input features are now not only restricted to one class of circuits, but to three. The features still correspond to the same four performance measures used in the Regression-only Model. The output layer is now not only comprised of a series of nodes that represent the circuit's sizes, but also an additional node for each class of circuits present in the dataset. The loss function used in the training of the networks will also be different, now taking into account both errors from the regression and the classification tasks. The weights assigned to each error measures are malleable, but weights of 70% and 30%, respectively, were used as a starting point.

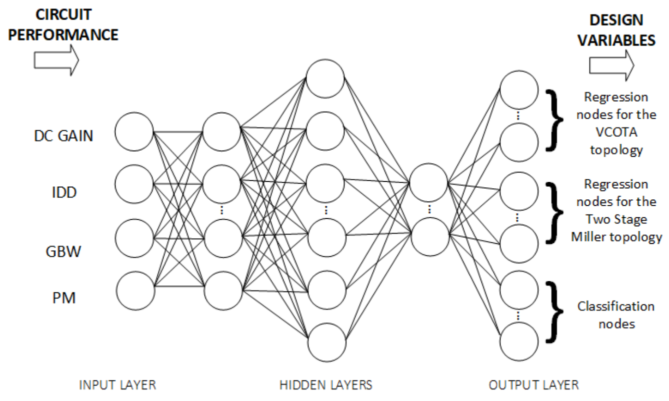

Fig. 5. Base structure of the Classification and Regression Model.

A. Polynomial Features and Data Normalization

Data preparation for this model involved the same steps of normalization and data augmentation through polynomial features as the ones from the Regression-only Model.

B. Model Structure and Hyper-parameter tuning

The guidelines that were used to select the hyper-parameters for the ANN Classification and Regression Model architecture were as follow:

- The number of input nodes and the number of hidden layers are the same as the Regression-only model architecture. The number of nodes in the output layer increases in relation to the previous model, which are now 30. This reflects the fact that the network is now processing different circuit performances and target circuit measures: 12 nodes for the VCOTA topology and 15 nodes for the Two Stage Miller Amplifier topology, and 3 additional nodes that encode the circuit class;
- The activation function used in all nodes (except in the output layer' nodes) is ReLU;
- Adam was the chosen optimizer, with learning rate = 0.001;
- Overfitting was addressed using L2 weight regularization, after the model showed to have a good performance;
- Initial random weights of the network layers were initialized by means of a normal distribution;
- Model performance was measured through a custom loss function (see expression (9)) that takes into account the error measurements from the classification nodes and from the regression nodes. Different percentages are assigned to each type of error, 30% and 70% respectively. Individual metrics were also used to prove the effectiveness of each task in the network. Regression error is calculated though a MSE function, while classification error is calculated through a Sparse Softmax Cross Entropy (SSCE) function;
- 5000 was the number of epochs chosen for initial testing. After having trained the first model, subsequent ANNs were trained with fewer epochs (500), using network weights from the ANN trained for 5000 epochs;
- A variable number was chosen for batch size, between 256 and 512, depending on the number of epochs;
- Finally, *gridsearch* is once again done over the hyper-parameters (number of layer, number of nodes per layer, non-ideality and regularization factor) to fine tune the model.

C. Training

The loss function, L, of the model that is optimized during training, is a weighted sum of two distinct losses – one from the regression task and the other from the classification task. Since this model's input features are not restricted to only one class of circuit performances, the regression loss will itself be a sum of the training errors from each circuit included in the dataset. Each individual regression loss is determined using MSE, like the previous model, while the classification error is measured through a SSCE function. This function measures the probability error in discrete classification tasks in which the classes are mutually exclusive (each entry is in exactly one class).

The loss function, L_{class}, that is optimized for the classification task is obtained by computing the negative logarithm of the probability of the true class, i.e. the class with highest probably as predicted by the ANN:

$$L_{class} = -\log p(\mathrm{TrueClass}) \tag{7}$$

The loss function, L_{reg}, that is optimized for the regression task is the MSE of predicted outputs Y' with respect to the true Y plus the L2 norm of the model's weights, W, times the regularization factor λ:

$$L_{reg} = \frac{1}{M}\sum_{j=1}^{M}\left(\left(Y'_{<j>} - Y_{<j>}\right)^{T}\left(Y'_{<j>} - Y_{<j>}\right)\right) + \lambda\left\|W\right\|^{2} \tag{8}$$

The total loss function, L, is the weighted sum between the two previous loss functions. Since there are two classes of circuits (excluding the third one, which is ignored in this function), there will be a distinct loss function value from each regression applied to each class. The MSE from each class is then multiplied by the true class predicted by the network in each step. This means that the MSE for the other class that was not predicted, will be neglected and become zero, i.e. if for a given step, VCOTA is the predicted topology, $\mathrm{TrueClass}_1$ will be greater than zero, while $\mathrm{TrueClass}_2$ will be equal to zero. The formulation of L is as follows:

$$L = 0.30 \times L_{class} + (0.70 \times (L_{reg1} \times \mathrm{TrueClass}_1 + L_{reg2} \times \mathrm{TrueClass}_1)) \tag{9}$$

The training of the models is again done using the Adam optimizer [12]. Other error metrics such as MAE and SSCE are also considered when validating the results. The results below are obtained for a model with one input and one output layer, and three hidden layers with 120, 240, 60 nodes each, for the demonstration of L2 regularization effectiveness.

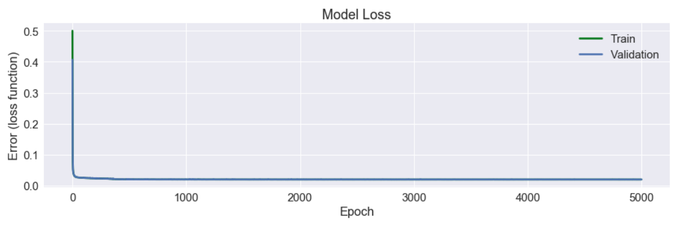

Fig. 6. Evolution of prediction error on train and validation sets during training, using L2 norm weigh regularization.

Similar to the previous architecture, model loss didn’t show overfitting after L2 regularization was included, as shown in Fig. 6.

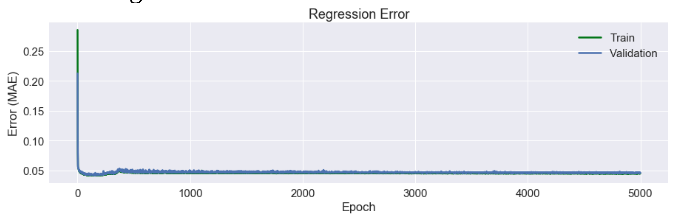

Fig. 7. Model regression error.

In this architecture, regression is performed using the same functions as the previous architecture: MSE for the training of the network and MAE for error measurement. Thus, the error obtained is similar to the one obtained in the Regression-only model, as shown in Fig. 7.

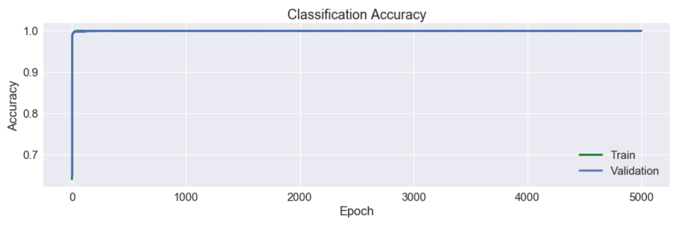

Fig. 8. Model classification error.

As shown in Fig. 8, classes from the all the design points are correctly predicted on the validation set.

VII. Case Study 2

For the second architecture, the two previously studied circuits were again considered. The goal of this architecture is to, not only learn design patterns from these circuits, but also to identify which circuit class can be sized by the set of input specifications. To do this, regression is applied to learn devices’ sizes and classification is used to learn circuit classes.

A. Dataset

For this example, the used dataset, i.e. Dataset-3, has 15,000 different design points. Each third of the dataset belongs to three different classes: the first class refers to circuit specifications that belong to the VCOTA topology (encoded as 001); the second class refers to circuit specifications that belong to the Two Stage Miller topology (encoded as 010); the third and final class is comprised of augmented data built up from the other two circuits (encoded as 100), but designed to not meet any of the specifications required by those circuits (i.e. maximum and minimum performance specifications are outside the required ranges). This last class was added to the problem to check if the network would be misled by input specifications that are out of the ranges required for the two studied circuits. The circuit performances that were considered to train the ANN were again DC Gain, IDD, GBW and PM, and, the ranges of values found in the dataset are shown in Table VIII.

TABLE VIII. Performance Ranges in the DataSet

		DC Gain	**GBW**	**IDD**	**PM**
VCOTA	***Max***	56.8 dB	78 MHz	395 uA	80º
	Min	44.7 dB	34 MHz	221 uA	60º
Two Stage Miller	***Max***	97.2 dB	102.8 MHz	0.8 uA	89.9º
	Min	59.8 dB	1.5 MHz	0.3 uA	55º
Augmented Data	***Max***	117.1 dB	33.2 MHz	0.3 uA	89.9º
	Min	69.7 dB	1.5 MHz	0.1 uA	55º

B. ANNs Structure and Training

Three ANNs were trained for this circuit, i.e., ANN-7 to ANN-9. The structure considered has 15 input variables (obtained from the second order polynomial feature extension of the 4 performance figures from Table VIII), 3 hidden layers with 120, 240, 60 nodes each, and, the output layer has 30 nodes, which represent the different devices’ sizes of the VCOTA and Two Stage Miller topologies (12 and 15 nodes, respectively) and the class to which they belong to (3 nodes to encode each of the three classes). For this dataset, the augmented data doesn’t have any devices’ sizes specified. Only performance figures were specified for this chunk of the dataset (as input features) so that a different class of circuits could be simulated.

ANN-7 was trained on the original dataset, for 5000 epochs with batches of 512 samples. Its training took less than 46 minutes. ANN-8 was trained on the augmented dataset, where 75K samples were generated for each circuit class, but only for 500 epochs. The network was initialized with weights from ANN-7. Its training took less than 50 minutes. ANN-9 was trained on the augmented dataset, where 100K samples were generated for each circuit class, for 5000 epochs. Its training took approximately 12 hours. Three performance metrics were used: a custom loss function (expressed in (9)), MAE for the regression nodes, and SSCE for the classification nodes. Their performance after training on the training and validation sets is summarized in Table IX.

TABLE IX. Performance of Trained ANNs for the Classification and Regression Model

	Loss Train	**Loss Val.**	**Train (MAE)**	**Val. (MAE)**	**Train (SSCE)**	**Val. (SSCE)**
ANN-7	0.0033	0.0034	0.0324	0.0329	1.0	1.0
ANN-8	0.0033	0.0033	0.0323	0.0324	0.9999	0.9999
ANN-9	0.0033	0.0033	0.0322	0.0321	0.9999	0.9998

Table X indicates the average matching rate between all the predicted and true devices’ sizes, and class prediction accuracy.

TABLE X. AVERAGE MATCHING RATE BETWEEN THE PREDICTED AND TRUE DEVICES' SIZES FOR THE CLASSIFICATION AND REGRESSION MODEL

Devices' sizes items		ANN-7		ANN-8		ANN-9	
		MAE	Class Acc.	MAE	Class Acc.	MAE	Class Acc.
VCOTA	_w8	7.51E-09	100%	6.64E-09	100%	6.08E-09	100%
	_w6	5.78E-06		5.82E-06		5.69E-06	
	_w4	2.38E-06		2.19E-06		2.21E-06	
	_w10	1.68E-06		2.22E-06		1.50E-06	
	_w1	1.27E-06		9.09E-07		8.71E-07	
	_w0	6.73E-06		6.92E-06		7.30E-06	
	_l8	1.71E-08		1.75E-08		1.54E-08	
	_l6	5.59E-08		5.64E-08		5.48E-08	
	_l4	5.13E-08		4.93E-08		4.94E-08	
	_l10	2.89E-08		2.76E-08		2.63E-08	
	_l1	4.26E-08		4.14E-08		3.89E-08	
	_l0	2.67E-08		2.45E-08		2.84E-08	
Two Stage Miller	_wb	1.18E-06	100%	1.03E-06	100%	1.05E-06	100%
	_wp	4.00E-06		3.64E-06		4.21E-06	
	_wal	1.57E-06		1.72E-06		1.65E-06	
	_w2g	3.34E-06		3.18E-06		3.82E-06	
	_lb	3.37E-07		3.05E-07		3.20E-07	
	_lp	2.31E-07		2.58E-07		2.68E-07	
	_lal	4.08E-07		3.82E-07		3.89E-07	
	_l2g	2.79E-07		2.47E-07		2.82E-07	
	_mbp	2.76E-01		2.64E-01		2.77E-01	
	_mb2	3.73E-01		3.39E-01		3.65E-01	
	_mal	6.75E+00		6.98E+00		6.72E+00	
	_mp	1.28E+01		1.00E+01		1.18E+01	
	_m2g	1.14E+01		1.10E+01		1.14E+01	
	_lc	4.65E-06		4.41E-06		4.78E-06	
	_nfc	9.44E+00		8.73E+00		8.87E+00	

In terms of performance, all three ANNs showed favourable results. From Table IX, we can observe that Loss, Regression and Classification errors were similar for all three networks. The MAE for each individual devices' sizes was also very similar across all ANNs.

C. *Sampling the ANNs for new designs*

From XI, we can conclude that ANN-9 can generate stable solutions for either topology, despite the considerable variability verified in certain specifications.

TABLE XI. PERFORMANCE OF SAMPLED DESIGNS

	DC Gain	GBW	IDD	PM	Topology
Target 1	50 dB	50 MHz	300 uA	60º	
ANN-9	54 dB	58 MHz	322 uA	57º	VCOTA
	54 dB	58 MHz	323 uA	58º	
	54 dB	58 MHz	322 uA	58º	
Target 2	70 dB	10 MHz	70 uA	60º	
ANN-9	83 dB	10 MHz	64 uA	59º	Two Stage Miller
	83 dB	10 MHz	63 uA	59º	
	83 dB	10 MHz	62 uA	59º	

VIII. CONCLUSIONS

In this work, deep learning methodologies were used to develop ANNs that successfully predicted analog IC sizing for an amplifier, given their intended target performances. This is a disruptive work, as no such approach has been taken in the field of analog and RF IC sizing, showing that a properly trained ANN can learn design patterns and generate circuit sizing that are correct for specification trade-offs, including those not provided in the training data.

To clarify, the purpose of this paper was not to propose a complete automation solution for the analog IC sizing, as this works only scratches the surface of the impact the ANNs and deep learning may have in analog CAD and EDA. There are still several opportunities where deep learning and ANN might improve analog EDA. A great possibility, and, at the same time one of the most challenging issues, is how to collect enough data to train such models. Given, of course, the importance of data, both in terms of quantity and quality, has in the train of the models. Data collections as aggregation is a great opportunity to the EDA community to define intra and inter-organization protocols and formats to create rich and meaningful datasets that can potentially enable true automatic analog design reuse. This is, reuses of the design patterns instead of specific solutions.

REFERENCES

[1] N. Lourenço, R. Martins, and N. Horta, “Automatic Analog IC Sizing and Optimization Constrained with PVT Corners and Layout Effects,” Springer, 2017. Hardcover ISBN 978-3-319-42036-3

[2] Cadence, “Virtuoso Analog Design Environment GXL”. Retreived from http://www.cadence.com, March, 2018.

[3] Mentor, a Siemens Business, “Eldo Platform”. Retreived from https://www.mentor.com/products/ic_nanometer_design/analog-mixed-signal-verification/eldo-platform, March 2018.

[4] N. Lourenço, et al., “AIDA:Layout-aware analog circuit level sizing with in-loop layout generation”, *Integration, the VLSI Journal*. 55(9):316-329, 2016.

[5] R. González-Echevarría, et al., “An Automated Design Methodology of RF Circuits by Using Pareto-Optimal Fronts of EM-Simulated Inductors,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 36, no. 1, pp. 15-26, Jan. 2017.

[6] Goodfellow, I., Bengio, Y., and, Courville, A. Deep Learning. MIT Press. 2016.

[7] G. Alpaydin, S. Balkir and G. Dundar, "An evolutionary approach to automatic synthesis of high-performance analog integrated circuits," in IEEE Transactions on Evolutionary Computation, vol. 7, no. 3, pp. 240-252, June 2003. doi: 10.1109/TEVC.2003.808914J.

[8] G. Wolfe and R. Vemuri, "Extraction and use of neural network models in automated synthesis of operational amplifiers," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 22, no. 2, pp. 198-212, Feb. 2003.doi: 10.1109/TCAD.2002.806600M.

[9] Hongzhou Liu, A. Singhee, R. A. Rutenbar and L. R. Carley, "Remembrance of circuits past: macromodeling by data mining in large analog design spaces," Proceedings 2002 Design Automation Conference (IEEE Cat. No.02CH37324), 2002, pp. 437-442. doi: 10.1109/DAC.2002.1012665

[10] Nobazaku Takai; Masafumi Fukuda, “Prediction of Element Values of OPAmp for Required Specifications Utilizing Deep Learning”, International Symposium on Electronics and Smart Devices, 2017, pp. 300-303, doi: 978-1-5386-2778-5/17/$31.00 ©2017 IEEE

[11] V. Nair and G. E. Hinton. “Rectified linear units improve restricted boltzmann machines”. In Proceedings of the 27th International Conference on International Conference on Machine Learning (ICML'10), 2010.

[12] D. P. Kingma, J. Ba, “Adam: A Method for Stochastic Optimization”. In: CoRR, abs/1412.6980, 2014.

[13] R. Povoa; N. Lourenco; R. Martins; A. Canelas; N. Horta; J. Goes, "Single-Stage Amplifier biased by Voltage-Combiners with Gain and Energy-Efficiency Enhancement," in IEEE Transactions on Circuits and Systems II: Express Briefs , doi: 10.1109/TCSII.2017.2686586

[14] M. Abadi, et al. “TensorFlow: Large-scale machine learning on heterogeneous systems”, 2015. Software available from tensorflow.org.

[15] Chollet, F., et al., “Keras”. GitHub, 2015.

[16] Fabian Pedregosa, et al. “Scikit-learn: Machine Learning in Python”, Journal of Machine Learning Research, 12, 2825-2830 (2011)

[17] John D. Hunter. Matplotlib: A 2D Graphics Environment, Computing in Science & Engineering, 9, 90-95 (2007), DOI:10.1109/MCSE.2007.55

[18] Stéfan van der Walt, S. Chris Colbert and Gaël Varoquaux. The NumPy Array: A Structure for Efficient Numerical Computation, Computing in Science & Engineering, 13, 22-30 (2011), DOI:10.1109/MCSE.2011.37