On the Exploration of Automatic Analog Integrated Circuit Placement using Neural Networks

Daniel Guerra
Instituto Superior Técnico - Universidade de Lisboa

***Abstract*—The work developed in this master thesis has the objective of automating a portion of the process of designing an analog integrated circuit, more specifically, the placement part of this process, where the position of devices are defined according to a set of topological constraints so that the minimum die area is occupied and the circuit's performance degradation from pre-layout to post-layout is minimized. This is one of the most critical parts of the whole circuit designing work flow and one of the most subjective as well, because each designer has his/her own preferences and layout styles when placing devices. This work proposes the construction of a model using neural networks that is based on previous placement designs and doesn't have to explicitly consider all the topological constraints when doing this process, since it should learn the patterns present in those legacy designs from circuit designers and/or solutions obtained by state-of-the-art automatic placement methodologies. The proposed model takes as input the sizing of the devices and outputs their coordinates in the circuit layout. The obtained results show that this model can not only replicate the legacy designs' placement, but also show indications that it learns patterns from different templates and apply them to new circuit sizings. In the case study in which the networks were trained with 12 different templates, the model has an average error of only $200nm$ per device and 97% of the placements it outputs have no overlaps.**

I. Introduction

The technological revolution in which we are living in brought the possibility of portable devices that are becoming smaller and lighter every day. Since these devices have to be supported by some kind of electronic circuit, these too have to follow the rule of becoming smaller as well, arising the need to combine both the digital and the analog circuits in the same chip to make a Mixed-Signal (MS) Systems-on-Chip (SoC). Although MS SoC have most of its area occupied by digital circuits, the analog part consumes an enormous effort by the circuit designer, as illustrated in figure 1.

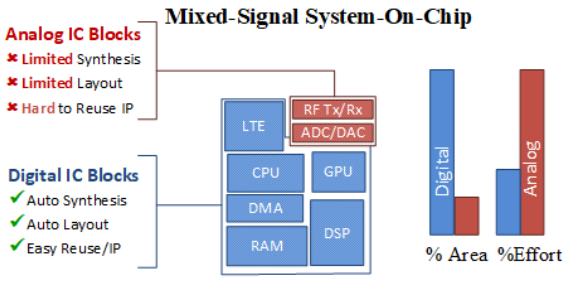


Figure 1: Comparison between the analog and digital sections of a MS SoC [1]

The development of automation tools for the design of analog circuits has been a very important topic to reduce both the manpower and the time needed to design a circuit.These tools are called Electronic Design Automation (EDA) tools and despite the importance of analog Integrated Circuits (ICs), there are not a lot of EDA tools established in the industry for analog circuits compared to their digital counterpart [2]. The lack of EDA tools for analog design makes it a bottleneck to the whole SoC designing process as Human intervention is necessary to search the solution space of both the sizing and the placement process.

Layout generation is part of a circuit design flow and corresponds to the task that both lays the devices, whose dimensions were previously determined for the selected topology, out in the chip and connects them [3]. It is common to split this whole process into two simpler problems: floorplanning and routing. In the floorplanning task, the devices are placed in the chip area and in the routing part of the problem the interconnections between devices are established. This first problem, which can be called placement, is the focus of this work and it is a complex task since there are several requirements that must be taken into account in order to reduce the unwanted effects of parasitic currents, process variations and different on-die operating conditions. In parallel to trying to satisfy these placement constraints, several objectives should also be minimized, such as chip area and interconnect length. These facts make the placement task very hard to automate, which made the automatic layout generation be intensively studied in the last three decades [4].

Research comunity proposed three different methodologies for automatic analog layout synthesis based on how legacy layouts are used in the design flow: layout generation considering placement and routing constraints, layout migration and retargeting and layout synthesis with knowledge mining. However, none of them succeeded in the industry, but recent developments in Artificial Intelligence (AI) and Machine Learning (ML) and their rising success in different fields of study, indicate that the knowledge mining route might bring better results for the automation of this process.

The main objective of this work is to develop a ML model that given the sizes of the devices of a circuit, it outputs their position in an IC layout to accomplish a speed up in the process of analog IC design, particularly in the placement part of the referred process. This model would be used to reduce the need of Human intervention, automating part of the process and, in conjunction with another model that would output

the sizing of the devices given the pretended performances, designers would have only to adjust minor details in the final circuit proposed by the models. Another appeal of using a ML system is the possibility of reusing all the layouts that were designed before. There are already a huge amount of analog layout designs stored that can be used as a baseline for new layouts. ML accomplishes this by using these legacy designs as training data to create a model.

This document is organized as follows:

Chapter 2 presents the state-of-the-art in analog IC layout synthesis, namely on the placement part of the layout generation process. After that, is given a ML overview of the most common techniques used in this field of study and the chapter is concluded with an explanation of how Neural Networks (NNs) work and some techniques that have to be taken into account when developing one.

Chapter 3 starts by discussing the circuit used for the tests, how the dataset is structured and the general architecture of the proposed NN. Moreover, the metrics used to evaluate the models are presented and some of the chosen hyperparameters to develop the networks are explained.

In chapter 4, the tests done to access the viability of using a NN to generate a layout placement are presented, as well as their results.

Chapter 5 presents the conclusion drawn from this work and outlines future developments that can be done to further optimize the performance of the methodology presented in this work.

II. Related Work

In recent years new EDA techniques and new analog layout synthesis methodologies have been introduced in order to increase how much the design process can be automated.

A. State-of-the-Art in Analog Integrated Circuit Layout Synthesis

There are three major methodologies used in modern analog layout synthesis, which differences lie in whether and/or how existing analog layouts are used during the process [2]:

- **Layout generation considering placement and routing constraints:** The devices are placed on the layout while respecting several placement constraints and achieving the minimal layout area. Usually this is done using an optimization-based approach. After that the interconnections between devices are established using a path-finding algorithm [2]. This methodology does not use any information from previously designed layouts. The disadvantage of this methodology is that every layout is generated from scratch and the result, is ultimately, unpredictable. The output is dependent of the amount of constraints set, and the result can be meaningful for the designer or not.
- **Layout migration and retargeting**: This methodology uses a previously designed analog layout of the same circuit and keep its topology for the new circuit with different sizes [2]. This is done by using layout compaction techniques, such as linear programming or graph-based algorithms [5] or using template-based approaches [6]. As major drawbacks, a single floorplan or legacy layout hardly yields compact placement solutions for the multitude of different sizing solutions that can be provided for the same circuit. Moreover, developing a placement template can be as time-consuming as designing manually the placement itself.
- **Layout synthesis with knowledge mining**: This approach uses a database of previously designed layouts [7, 8]. It uses data mining techniques to extract matched sub-circuits between the new and legacy designs and then produces new layouts using those matched sub-circuits [2]. Systems that fall into this methodology use a graph-based approach [8], which makes them deterministically solve the conflicts on the graphs to construct a new layout. Their main drawback is that they cannot effectively deal with high amounts of circuit data, as it does not generalize the acquired ”knowledge”.

Determining the location of each device or device group in the actual layout is one of the most critical steps in analog layout synthesis because of all the constraints that have to be satisfied, since devices interfere with each other in ways that may affect the circuit performance. Furthermore, since the placement task precedes the routing generation and has a definitive impact in the attainable interconnect quality, most of parasitic effects and consequent post-layout circuit's performance degradation are set once a placement is fixed. According to [2], recent studies address eight placement constraints to take into account during the analog placement step. They are the symmetry, symmetry island, common centroid, proximity, regularity, boundary, current/signal path and thermal constraints. They all restrict the position of a device or device group to minimize the wirelength or to achieve a better performance.

This amount of constraints makes the decision of where a device should be located a very complex task. The methodology proposed in this work can abstract the need to follow these constraints by learning patterns from legacy designs in which they were already respected, if previously considered relevant by the layout designer or EDA engineer.

B. Contributions: Machine Learning and Analog Layout Placement

This work proposes the development of a NN to model the process of placing devices given its sizes. The objective of using a ML algorithm is that it is possible to abstract from all placement constraints by using previously layouts that compose the dataset, assuming that those layouts were approved by the designer and/or were used for the creation of a circuit.

Before choosing NNs as the ML method to use in this work, different methods were compared. It is possible to divide all the ML in five different ”tribes” [9]: symbolists, bayesians, connectionists, evolutionaries and analogizers. The

main algorithm of each tribe was explored in order to conclude which one was best suited for this work.

Symbolists' Decision Trees (DTs) are easy to interpret, can easily handle feature interactions which makes them insensitive to outliers and are non-parametric which makes them easy to implement. Random forests attenuate the fact that DTs are easy to overfit and they are fast and scalable and require a low amount of tuning, however, these algorithms are not the most suitable for more complex problems where several rules must be inferred from the data.

Bayesians' algorithm is well-suited for classification problems, but not so much for the regression ones, that is the case of the placement problem, which makes algorithms like Naive Bayes not an optimal choice.

Evolutionary algorithms, like Non-dominated Sorting Genetic Algorithm - II (NSGA-II) [10], can solve problems the can be represented as a optimization task [11], which is the case of the placement process. These algorithms are already used to optimize the currently used placement methodologies of absolute and topological representations described in II-A. However, this work proposes a different approach in order to reuse legacy layout designs.

Although analogizers' Support Vector Machines (SVMs) have their main use in classification problems, they can also be used to solve regression problems. However, this algorithm has a high memory complexity, the obtained results are somewhat hard to interpret and they are very hard to tune since it may be hard to find a suitable kernel function to solve a problem.

The development of this work led to the creation of a NN that outputs a possible placement for a circuit based on the sizing of its devices. This approach abstracts the need of following complex and conflicting placement requirements by following the pattern of legacy designs that already satisfied them. The advantage of using a NN is that it predicts a placement for a given sizing at a push button speed, reducing the time spent in this task of the whole IC design process. The proposed NN also output three different placements (one with the smallest layout area, one with the minimum Aspect Ratio (AR) and one with the maximum AR) to be used depending on the situation.

From the best of our knowledge, this is the first exploratory work in this research field that combines the potential of AI, and more specifically, NNs, to improve analog IC placement automation. The ultimate objective of this research is to lead to a model that would be trained with different circuits and could predict the placement for a sizing of any circuit topology (within certain classes of circuits and with a maximum number of devices). It is important to note that the architecture of the network must be changed to accept different sized inputs, as each circuit can have a different number of devices. The way these circuit topologies are encoded in the input layer of the NN is an important subject in analog EDA research community and it is left for future research directions [12].

III. Development of a Neural Network for Analog IC Placement

A. Circuit used for tests

The circuit chosen to develop the NN was the single stage amplifier. Its schematic is shown in figure 2. This circuit was chosen due to its moderate complexity, as it contains 12 devices grouped into 6 symmetry pairs, whose layout locations must be predicted.

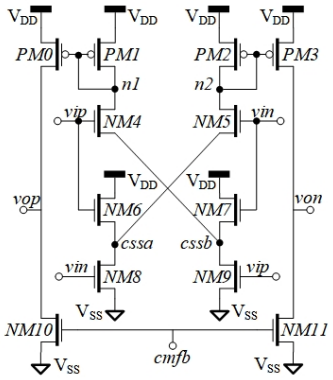


Figure 2: Single stage amplifier schematic [13].

There are different possible placement solutions for the same circuit. This possibilities can be represented as templates that represent the relative positions among devices. This relations are kept even for different sizes. Examples of placement templates for the single stage amplifier are shown in figure 3.

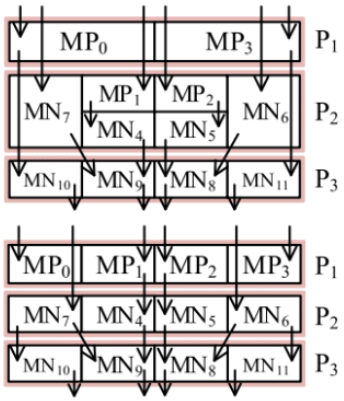


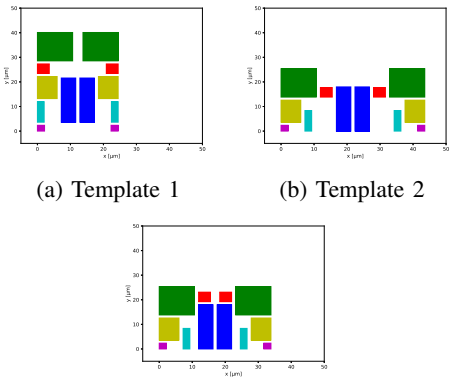
Figure 3: Two different handmade placement templates with current-flows highlighted [13].

B. Dataset Architecture

The measures of the sizing of the devices used as the inputs of the NN that were chosen were the transistor's total width, length and number of fingers and the total width and height of the respective parametric cells, i.e., the measurements for the layout implementation of each device. The width and height of the devices' layout implementation were obvious choices because they define the area it occupies but the remaining parameters of the devices define their properties.

To measure the position of the device in the IC various measures can be used such as the Cartesian coordinates of a corner of the device or its center. Furthermore these positions can be relative to the corner of the IC or to the axis of symmetry that must be respected because of the symmetry constraint of the placement phase.

A single sizing of the devices can produce different valid placements. To take that into account, the dataset contains placements of the same sizing for 12 different templates, simulating the existence of placement examples from different designers. The resulting layout placement for the same sizing according to three different templates is shown in figure 4.



(a) Template 1 (b) Template 2

(c) Template 3

Figure 4: Layout placement for the same sizing, but for different templates

The dataset using to train the NN only contains the placement for 3 layouts (smaller area, biggest and smallest AR) including the index of the template that produced that layout, because for a given sizing, a template might lead to a small area layout but for other sizings it will not. As the original dataset contains the area, total height and total width of the IC layout, it has to be processed to compute for each sizing example, the template that has produced the layout with smaller area and biggest and smallest AR. The dataset for the single stage amplifier circuit contains around $10,000$ where 80% was used to train the network and the testing set contained 20% of this dataset.

A section of the dataset is presented in tables I and II. In the first the columns that correspond to the input of the model are presented and the latter corresponds to the columns that contain the placement information for all templates. The transistors' total width, length and number of fingers correspond to the columns starting with $w\_$, $l\_$ and $nf\_$, respectively. The width, height and area of a layout are represented in columns $width$, $height$ and $area$, respectively. The remaining columns starting with $x\_$ and $y\_$ contain the coordinates of the bottom left corner of each device in the corresponding template. The dataset used to train the network follow the same rules, but is only contains three templates: the one with smaller area, the one with maximum AR and the one with minimum AR

Table I: Section of the input columns belonging to the original dataset with all templates.

Device 1					...	Device 11				
w_0 [μm]	l_0 [μm]	nf_0	wt_0 [μm]	ht_0 [μm]	...	w_11 [μm]	l_11 [μm]	nf_11	wt_11 [μm]	ht_11 [μm]
17.5	0.36	3	3.48	8.07	...	8.7	0.89	3	4.73	5.14
17.7	0.36	7	6.52	4.765	...	83.3	0.83	7	9.47	14.14
50.7	0.40	1	2.0	52.06	...	20.8	0.77	3	4.37	9.17

Table II: Section of the column with the placement information belonging to the original dataset with all templates.

Template 1								...	Template 12							
width [μm]	height [μm]	area [μm^2]	x_0 [μm]	y_0 [μm]	...	x_11 [μm]	y_11 [μm]	...	width [μm]	height [μm]	area [μm^2]	x_0 [μm]	y_0 [μm]	...	x_11 [μm]	y_11 [μm]
20.17	101.77	2052.7	10.71	4.01	...	15.44	4.01	...	24.23	60.46	1464.9	6.69	6.39		6.69	0.0
35.73	98.79	3529.8	18.49	4.01	...	26.26	4.01	...	33.57	59.69	2003.7	6.69	15.39		6.69	0.0
23.15	109.0	2523.5	12.2	3.99	...	18.78	3.99	...	28.79	132.79	3823.0	9.4	10.42		9.4	0.0

Feature engineering is an essential part of the creation of a ML model to make modifications on the features of the model so that better results are achieved or eliminating features that are not important. Adding polynomial features is an example of the first. This process increases the number of inputs in which the new features correspond to the polynomial built from the existing features. For example, if there are features a, b and c, the polynomial features of the 2^{nd} degree are: a, b, c, a^2, ab, ac, b^2, bc and c^2.

The increase in the number of inputs opens the possibility of making the model learn new patterns that are only present with polynomial combinations of the input variables, for example, in this work it makes sense to have the device area as an input, which is accomplished by the addition of polynomial features since it corresponds to the multiplication of the width feature by the height feature.

C. Neural Network Architecture

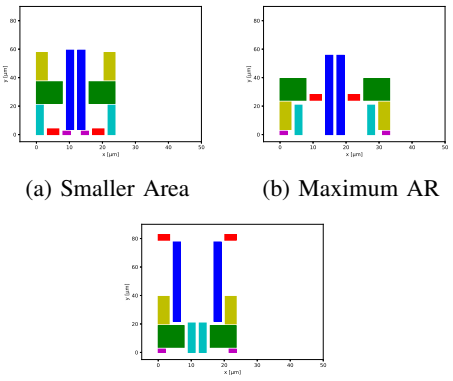
This work proposes the development of a model that predicts the placement of all the devices of an analog IC based on their sizings. The proposed model is a non-linear model described by a NN. To achieve the intended results, the inputs of the model have to be some measure of the size of the devices and the respective output have to be the actual position of the devices on the IC. Since the number of inputs and outputs vary with the number of devices in the circuit, each model can only describe a specific circuit or at most circuits with the same number of devices.

In order to offer more possibilities of IC layouts, the output of the NN described above can be triplicated so that the theoretical output is the placement for 3 different layout categories: the one with the smaller area and the ones with biggest and lowest AR, for a more horizontal and vertical layout, respectively. Layout placements for these three categories are illustrated in figure 5 for the same circuit sizing.

A general network that represents the described architecture is represented in figure 6. The number of neurons in each layer is variable, as well as the number of inputs layers.

D. Pre-Processing the Data

Pre processing the dataset is a fundamental step of any ML process and it is strongly dependent on the problem and what



(a) Smaller Area (b) Maximum AR

(c) Minimum AR

Figure 5: Layout placement for the same sizing, but for different layout categories.

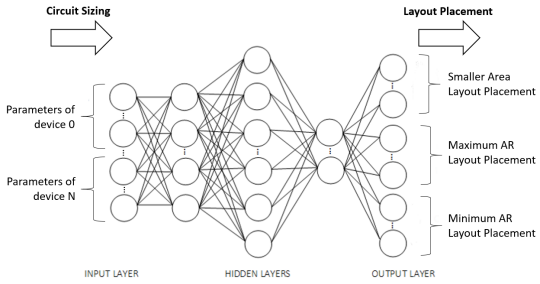


Figure 6: Network architecture used to predict layout placements given the circuit sizing.

the dataset represents.

In this work, the dataset contains sizes of devices and their respective localization. Since the electronic devices used in ICs are very small, they also result in small circuits. This fact causes the values present in the dataset to be very small (in the order of the nanometers) and will cause the gradients computed in the backpropagation method to be even smaller, to a point where a computer might not have bits to represent it and so the updates on the weights of the network will be wrongly computed. To solve this problem it is common to scale the dataset so that the mean of each column of the dataset is 0 and the variance is unitary using the equations 1, 2 and 3, where z is the scaled column of the dataset, x is the unscaled value of the same column, μ is its mean and σ is its variance. There is also the possibility of scaling each column of the dataset so that its values are between 0 and 1, following equation 4, where x_{min} and x_{max} represent the minimum and the maximum value of the unscaled column. This method of scaling scales the maximum value to 1, the minimum value to 0 and every value in between them to be proportionally scaled between 0 and 1.

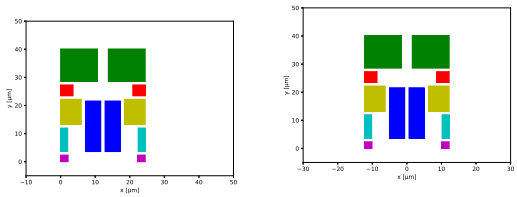
$$z = \frac{x - \mu}{\sigma} \tag{1}$$

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{2}$$

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2} \tag{3}$$

$$z = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{4}$$

Besides scaling, it is also a good practice to transform the data in some way that makes it easier for the algorithm to learn patterns in the problem. The proposed outputs of the model are the coordinates of the bottom left corner of each device, assuming the bottom left corner of the IC is at $(0, 0)$. However since analog ICs are symmetric, it is possible to assume that the symmetry axis is at $x = 0$ and make the output of the model be symmetric values that represent the position of the devices, like the center point of each device or a left corner for the left-side devices and the right corner for the right-side devices.



(a) Original representation (b) Representation after centering data

Figure 7: Representation of the placement of the devices before and after centering the data

E. Post-Processing the Data

Usually regression NNs do not have perfect accuracy and so the output might deviate from the expected values. To correct this limit of these networks, their output is usually post-processed so that the overall output of the model is adequate to solve the problem it is projected to solve. In the case of this placement problem the output of the network might be a placement where pair of devices are not exactly symmetric or, in the worst case, where devices are overlapped. To ensure those issues do not occur it is possible to design an algorithm that slightly changes the coordinates of each device so that no device is overlapped with another and then change their position again so that each pair of devices is symmetric.

F. Metrics to evaluate the models

In order to evaluate a trained model, one has to develop some kind of metric to compare different models. The model is evaluated using a different set (denominated test set) than the one used to train the model, being data the model never seen since the objective is to conclude how well the model can generalize.

1) Error related metrics: The more immediate metric is an error metric, like the Mean Squared Error (MSE) and the Mean Absolute Error (MAE). This metrics measure how much the placement predicted by the model differs from the target placement present in the dataset. To measure this error there are different quantities that can be computed that have different meanings, like the MSE, MAE and the Mean Absolute Error per Device (MAED). These metrics are computed as shown in equations 5, 6 and 7, respectively, where N is the number is the number of examples in the test set, M is the number of devices in the circuit, x_{ij}^p is the predicted x coordinate of device j in example i and x_{ij}^t is the target x coordinate for the same case. The MSE is used to train the model since it is easily differentiable, however it is not a good measure to represent a real error, since it corresponds to a squared error. The MAE, however, represents the error of the placement across all devices (in the units of the output) that is expected when the model predicts a placement. The MAED indicates how much the placement of a device differs from the target placement on average. Since both the MAE and the MAED represent a error in the dimensions of the output variables, they are easy to evaluate metrics.

$$MSE = \frac{1}{N}\sum_{i=0}^{N}\sum_{j=0}^{M}[(x_{ij}^p - x_{ij}^t)^2 + (y_{ij}^p - y_{ij}^t)^2] \quad (5)$$

$$MAE = \frac{1}{N}\sum_{i=0}^{N}\sum_{j=0}^{M}\sqrt{(x_{ij}^p - x_{ij}^t)^2 + (y_{ij}^p - y_{ij}^t)^2} \quad (6)$$

$$MAED = \frac{1}{NM}\sum_{i=0}^{N}\sum_{j=0}^{M}\sqrt{(x_{ij}^p - x_{ij}^t)^2 + (y_{ij}^p - y_{ij}^t)^2} \quad (7)$$

2) Overlap related metrics: Although computing the error of a prediction is a valuable metric, in this specific problem, if the error of a prediction is big, it does not mean that the predicted placement is wrong, since it can be a valid circuit and might even be a better placement than the target. To evaluate this cases there is a need to develop a metric that evaluates if the circuit is valid or not.

In this sense, one can compute the total overlap area among all the devices, because the model might output a placement where this occurs and it is a clear indicator of a wrong placement.

Computing the total overlap area for each example it is then possible to compute the Mean Overlap Area (MOA) over a set of examples. This is the metric used to evaluate the amount of overlap that the model outputs.

3) Accuracy: Since this problem is in its core a regression problem there is no direct way of knowing if the model output is correct or not. To have a metric to represent the accuracy of the model, it was considered that it was 'correct' if the error or the overlap were behind a reasonable threshold, since if they were small, a post processing algorithm could eliminate those small deviations from the correct answer. With these metrics it is possible to compute the percentage of cases where the model gave an output in which the error or the amount of overlap can be considered 'correct', Error Accuracy (EA) and Overlap Accuracy (OA), respectively. To count the cases where both the error and the overlap of the output placement were minimal, it is possible to compute the Error and Overlap Accuracy (EOA).

G. Training of the Network

To train a network and build one that has the optimal performance, it is necessary to tune a lot of different hyperparameters. This is the most difficult process of this models as its an iterative process and it can be a prolonged process since training different networks to compare performances can take a long time. In order to turn this process easier, in this work, some hyperparameters were set to default values that achieve good performances is almost all situations. This were the case of the activation function and the hyperparameters corresponding to the optimizer used to find the minima of the cost function.

The activation function in the hidden layers was the Exponential Linear Unit (ELU) function and the linear function was used in the output layer. The optimizer used was the Adaptive Moment Estimation (Adam) optimizer and its parameters were set to the default values.

The remaining hyperparameters were set by trial and error or, in the case of the batch size, to a value that makes sense. The batch size was set to 500 as it is a value that it is not too small nor too big. Since the training dataset contains around 8000 examples, 500 is around 6.25% of the dataset which is lower than the 10% that is considered a big batch size. The downside of this lower value is that the training times get higher, while requiring less epochs to achieve convergence. The number of epochs and the number of neurons were set by trial and error by training different networks and comparing their performances. The results of these tests are shown in section IV.

The development of the NN with an architecture that achieve a better performance will be iterative. First, a network with no hidden layers will be tested to access if a linear regression can be enough to bring good results. The hyper parameters of the network will be tuned to check its effect on the performance. Then, polynomial features will be added to check its affect on the performance of the model.

After that, simpler networks with less hidden layers will be tested first and then their depth will be gradually increased until the performance is acceptable or until the training times and/or memory needed become too big. Centering the data might make it easier for the algorithm to learn the patters on the data and so it will be applied to the dataset and it will be checked if the results get better. In case of overfitting, a regularization technique method will be applied to network in order to attenuate it.

IV. Experimental Results

After testing different network architectures, it was concluded that polynomial features of the 2^{nd} degree are essential

to achieve a good performance. Higher degree polynomial features of higher orders were not tested due to the lack of computational resources. It was also concluded that 3 and 4 hidden layers are necessary to achieve acceptable performances. Shallower depths led to worst results and deeper networks were not tested due to lack of computational resources, as well. In this regard, the network architectures that led to better results are presented in table III. The networks were trained during 1000 epochs.

Table III: Attributed names to each NN architecture

		Number of Neurons			
Name	Hidden Layers	Layer 1	Layer 2	Layer 3	Layer 4
NN-3	3	1000	500	250	
NN-4	4	1000	500	250	100

The thresholds used to verify the correctness of the proposed placement by the model using the EA and the OA were $100nm$ and $0\mu m^2$ (meaning only placements with no overlap are counted as correct), respectively.

A. Centering Data

To try to make the patterns between the input and the output more evident, the dataset can be pre-processed before training the model, so that the symmetry axis is considered the $x = 0$ of the placement. This will make the outputs that represent the x coordinates of pair of devices symmetrical, which might make the patterns in the data to be easier for the network to learn. The results are shown in table IV

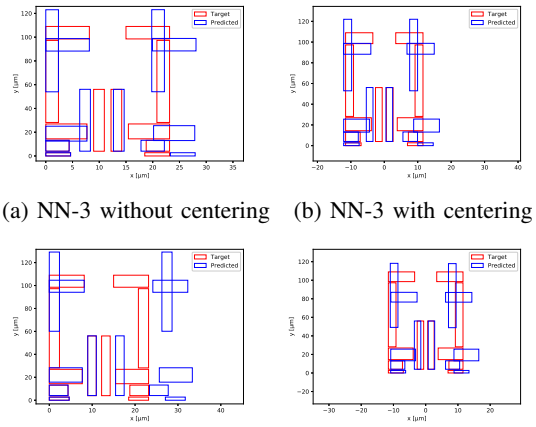
Table IV: Performance of trained NNs with and without centering the data

		Test				
NN	Centering	MAED (nm)	MOA (μm^2)	EA (%)	OA(%)	EOA (%)
NN-3	No	167.502	0.029	46.689	99.472	46.689
	Yes	65.513	0.044	94.146	99.808	94.146
NN-4	No	55.295	0.036	93.282	99.664	93.282
	Yes	79.068	0.035	85.893	99.664	85.893

Only NN-3 got an improvement in the when the data was centered. NN-3 with centering and NN-4 without centering seem to be the networks that achieve a better performance, differing in MOA for about 1% in the test set, favoring NN-3, however, NN-4 has a lower MAED. These are the only architectures that achieve a MOA superior to 90%, so they will be the only ones that will be considered in further tests. In figure 8, it is possible to observe that the placement with maximum error output by NN-3 and NN-4 is close to the target and with little post-processing, the difference can be solved. It is also possible to observe a minimal improvement in relation to the models trained without centered data.

B. Multiple Templates

Since there can exist layouts based on more than one template on the database, the dataset was modified so that, for each sizing, the template that led to a placement with smaller layout area was chosen.



(a) NN-3 without centering (b) NN-3 with centering

(c) NN-4 without centering (d) NN-4 with centering

Figure 8: Target and predicted placement for the examples with largest error tested on every named NN trained without and with centered data.

For these tests, the datasets were created with different number of templates, choosing the best layout (with smaller area) among that number of templates. As some templates are better for certain sizes, the dataset will contain examples where the solution corresponds to a template and other examples that correspond to another. This fact makes so that the patters in the data are harder to identify by the algorithm. In this regard, the number of epochs was increased to 1500, since the optimization algorithm takes more time converging due to the conflicting different templates. The network was trained with four datasets with different number of templates: 2, 4, 8 and 12. The results for the networks trained with 4 and 12 templates are presented in tables V and VI, respectively.

Table V: Performance of trained NNs with and without centering the data, when the dataset has 4 different templates

		Test				
NN	Centering	MAED (nm)	MOA (μm^2)	EA (%)	OA(%)	EOA (%)
NN-3	No	558.915	3.761	6.622	92.850	6.622
	Yes	429.202	3.574	65.211	93.138	65.211
NN-4	No	537.602	3.168	29.511	91.747	29.511
	Yes	439.921	3.033	55.566	95.345	55.566

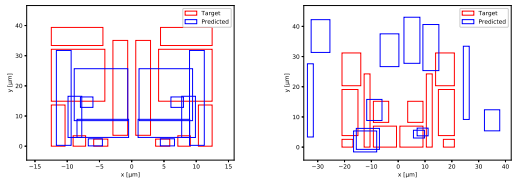
Table VI: Performance of trained NNs with and without centering the data, when the dataset has 12 different templates

		Test				
NN	Centering	MAED (nm)	MOA (μm^2)	EA (%)	OA(%)	EOA (%)
NN-3	No	453.008	2.215	0.0	93.106	0.0
	Yes	206.533	0.476	21.913	97.393	21.913
NN-4	No	459.814	1.897	0.352	89.747	0.352
	Yes	497.691	3.359	2.079	85.333	2.079

Since the existence of different templates in the dataset makes the patterns in the data to be more difficult to learn, it is expected that the performance of the system decreases relative to the previous one where only one template was contemplated. NN-3 with centering achieves a better performance in almost all tests. As the number of different templates increases

the performance of the model starts to decrease as expected, due to the variation in patterns it tries to learn. The models trained with 8 and 12 templates have a low EOA which makes these architectures not so well-suited for real life application as their output will not be optimal.

Even though the output of the model is not optimal it still outputs a placement that is close to a valid one in the sense that it almost symmetric as seen in figure 9a. Although, the overlap in this case is high, it is possible to conclude that it is trying to follow the patterns of a template, being almost symmetric. This fact means that the network learned the patterns of a template or subgroup of templates and applied to this example. However, there also cases with high error that do not follow any rules like the predicted placement shown in figure 9b. The high values of error can just mean that the network over-learned a template and applied those rules to a sizing which placement respected a different template.



(a) 12 templates (b) 8 templates

Figure 9: Target and predicted placement for the examples with largest error tested on NN-3 with centered data and with different number of templates.

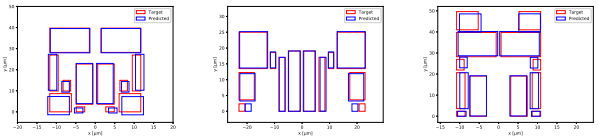
C. Multiple Placement Outputs

To make the ML system proposed in this work more useful in real-life applications, the output of the model was changed so that it would produce 3 different outputs: one with the smallest area, one with the maximum AR and one with the minimum AR.

The distribution of each template in the different datasets used to train the network is presented in table VII. Since there are datasets where the maximum and/or minimum AR outputs almost only contain one template it is the same as training a network with a single template and, for that reason, it is expected that the accuracy, in this cases, is high.

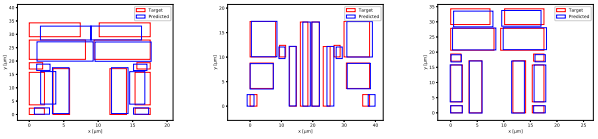
The only networks that were trained to check the results of this output architecture were NN-3 with centering and NN-4 without centering, as they were the network architectures that achieved better results when trained with datasets that contained more than one template. The results are shown in VIII and IX. NN-3 almost always outperforms NN-4 specially when the number of templates in the dataset starts to increase.

Although this models have a hard time learning patterns, not only due to the high number of templates, but also due to the fact that the weights of the network are shared for the 3 proposed placements, they still output valid (or close to valid) layout placements. Figure 10 illustrates this, as it corresponds to the median of the error for the NN-3 with centered data and trained with 12 templates. In the case of NN-4 without centering the data but trained with 12 templates, the same happens as it is shown in figure 11. Even though the complexity of the dataset, these models can still output a close to valid placement that can easily be post-processed to make it closer to the target placement.



(a) Smaller area (b) Maximum AR (c) Minimum AR

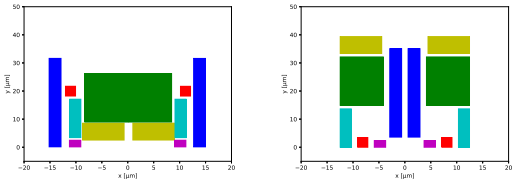
Figure 10: Target and predicted placement for the example corresponding to the median error. Used model was the NN-3 with centered data and 12 templates.



(a) Smaller area (b) Maximum AR (c) Minimum AR

Figure 11: Target and predicted placement for the example corresponding to the median error. Used model was the NN-4 without centered data and 12 templates.

The accuracy values starts to decrease, when the number of templates used to train the network increases, as happened in the previous test with multiple templates but just one placement output. This was expected again, but the higher values of error do not mean that the placement output by the network is not valid. As seen in figure 12, the predicted placement shown in figure 12a is very different from the target (12b), hence the high error, but might be a valid one as it has no overlap and it is symmetric. This difference on the output might be caused by the fact that the network strongly learned the patterns of one or more templates and then applies them to every input.



(a) Predicted placement (b) Target placement

Figure 12: Predicted and target placement of the example with largest error for the layout with smaller area. The model used was the NN-3 with centering.

Table X shows the presence of each template in each dataset (training and testing) and the accuracy achieved in examples of

Table VII: Distribution of each template in the datasets.

	Number of Templates in the Dataset											
	2			4			8			12		
Template	Smaller Area (%)	Maximum AR (%)	Minimum AR (%)	Smaller Area (%)	Maximum AR (%)	Minimum AR (%)	Smaller Area (%)	Maximum AR (%)	Minimum AR (%)	Smaller Area (%)	Maximum AR (%)	Minimum AR (%)
1	10.662	0.0	100.0	3.826	0.0	42.264	0.552	0.0	40.705	0.116	0.0	37.091
2	89.338	100.0	0.0	8.887	99.964	0.0	0.048	3.154	0.0	0.036	0.116	0.0
3				32.058	0.036	0.348	8.527	0.012	0.300	4.845	0.0	0.068
4				55.229	0.0	57.388	37.635	0.0	57.064	37.683	0.0	57.660
5							14.824	0.0	1.931	14.276	0.0	0.816
6							16.347	96.726	0.0	12.101	0.0	0.0
7							18.781	0.108	0.0	8.971	0.0	0.0
8							3.286	0.0	0.0	3.350	0.0	0.0
9										0.264	0.008	0.484
10										0.272	69.957	0.0
11										0.0	29.919	0.0
12										18.086	0.0	3.882

Table VIII: Performance of NN-3 with centered data across the 3 placement outputs, when different number of templates are present in the dataset.

	Test								
	Smaller Area			Maximum AR			Minimum AR		
Templates	MAED (nm)	MOA (μm^2)	EOA (%)	MAED (nm)	MOA (μm^2)	EOA (%)	MAED (nm)	MOA (μm^2)	EOA (%)
1	58.103	0.068	90.355	57.981	0.056	90.499	58.058	0.056	90.307
2	238.624	0.172	81.574	90.495	0.044	87.716	105.242	0.053	82.006
4	467.690	4.500	57.821	91.599	0.071	85.797	404.523	4.145	47.073
8	893.245	10.538	16.891	248.068	1.784	79.271	434.566	5.330	52.783
12	297.888	1.256	11.548	300.019	0.443	6.878	206.100	0.528	31.078

Table IX: Performance of NN-4 without centered data across the 3 placement outputs, when different number of templates are present in the dataset.

	Test								
	Smaller Area			Maximum AR			Minimum AR		
Templates	MAED (nm)	MOA (μm^2)	EOA (%)	MAED (nm)	MOA (μm^2)	EOA (%)	MAED (nm)	MOA (μm^2)	EOA (%)
1	61.193	0.032	91.603	61.199	0.032	91.603	61.188	0.032	91.603
2	235.945	0.142	83.301	110.155	0.068	86.756	96.260	0.055	85.413
4	457.402	3.583	48.417	130.160	0.052	81.574	411.164	3.526	41.315
8	887.761	8.640	26.008	322.172	1.551	67.658	454.046	3.118	46.113
12	562.028	3.849	0.016	603.367	1.949	0.304	404.998	1.179	0.240

each template. As expected it can be seen that the percentage of a template in the training dataset is correlated to the accuracy on the examples of that template, as the greater the percentage is, the greater is the EOA. This comes from the fact that the network was trained with more examples of a template and so, it learned those patterns better.

D. Discussion

The introduction of different templates in the dataset affects negatively the performance of the models if its evaluation is done by comparing the error values, however the presence of diverse templates makes the network over-learn the patterns of some of them and then apply them to other inputs. In these cases the output has high error, since the target and predicted output are following different templates, but it is a valid layout placement.

The final model with 3 proposed layout placements has a good performance and it gives the user a possibility of choosing the layout configuration that is best for the application he/she wants.

Conclusions and Future Work

This chapter presents the conclusions of this work, as well as future directions for further applications of NNs to the placement process of analog layout design.

V. Conclusions

This worked presented ML methodologies to be used on the placement part of the layout generation process. Different NNs with different architectures and datasets were developed and although some of them that tried to solve more complex problems (presence of many templates in the dataset) did not achieve a good performance according to the proposed metrics, it was shown that they would output a valid layout placement. However, it would output a high error since it did not follow the rules of the supposed template, which may be caused by the fact that those networks learned the patterns of the different and conflicting template guidelines, and applied them to the inputs. Some of these layout placements can even occupy a smaller area than the one presented in the dataset, however, it might not be valid due to the device-to-device interactions.

With all things considered, NNs seem to be suited for this problem as they are quite effective in simpler datasets with a low amount of different templates and seem to learn the rules to output a symmetric layout placement.

A network with 3 hidden layers seems to be the most optimal for this problem and it achieved an acceptable average error of only $200nm$, where in 97% of the cases, it outputs a placement with no overlaps. Some cases with high error, were also close-to-valid placements, in the sense they were almost symmetric, but the network applied the patterns of a template that was not the same as the target. There are cases where the output layout area is even smaller than the target. To verify the validity of these cases, it would be needed to simulate the layout after applying a routing algorithm to verify if it achieved the desired performances.

This work proved that NNs can learn the intricate patterns on the data and can be helpful in the analog layout synthesis process by speeding up the layout generation process and with further developments might be used in real-life applications, thus decreasing or eliminating the manpower needed for this task.

VI. Future Work

The development of this work fell short in overfitting the networks, since regularization techniques did not increase the performance of the models. This may have happened due to the lack of experimentation on deeper and more complex networks. However, the development, test and tuning of these kind of networks is time-consuming and could only be considered a valid theory after the work presented in this master dissertation is validated. In this regard it would be interesting to test more complex architectures to verify if better performances can be obtained and then apply a regularization technique such as dropout.

To ensure that the output of the network makes sense, it would be necessary to apply a deterministic post-processing

Table X: Presence in each dataset and accuracy of each template.

	Smaller Area			Maximum AR			Minimum AR		
Template	Presence in Testing Dataset (%)	Presence in Training Dataset (%)	EOA (%)	Presence in Testing Dataset (%)	Presence in Training Dataset (%)	EOA (%)	Presence in Testing Dataset (%)	Presence in Training Dataset (%)	EOA (%)
1	0.160	0.116	0.0	0.0	0.0	-	37.876	37.091	21.284
2	0.0	0.036	-	0.064	0.116	0.0	0.0	0.0	-
3	5.326	4.845	0.0	0.0	0.0	-	0.016	0.068	0.0
4	37.716	37.683	21.416	0.0	0.0	-	56.734	57.660	40.569
5	14.571	14.276	1.207	0.0	0.0	-	0.912	0.816	0.0
6	12.332	12.101	1.167	0.0	0.0	-	0.0	0.0	-
7	8.541	8.971	0.562	0.0	0.0	-	0.0	0.0	-
8	2.767	3.350	0.0	0.0	0.0	-	0.0	0.0	-
9	0.336	0.264	0.0	0.016	0.008	0.0	0.416	0.484	0.0
10	0.208	0.272	0.0	70.250	69.957	6.444	0.0	0.0	-
11	0.0	0.0	-	29.671	29.919	7.925	0.0	0.0	-
12	18.042	18.086	17.199	0.0	0.0	-	4.047	3.882	0.0

to it. This post-processing would ensure that there any devices overlapped and that the circuit is completely symmetric. It would also be useful to apply a routing algorithm to the layout placement proposed by the network and then simulate it to see if it is a valid circuit to achieve the desired performances or if it should be discarded.

An algorithm that would evaluate how similar a placement is to each template would be appropriate to access if the network learned to replicate a subset of templates or if it actually learned the best patterns from each template and then applies them to the input data.

Another interesting hypothesis is to have a graph representing the circuit as an input to the network using a Graph Neural Network (GNN). This would give the network the relation between devices which might help the learning process.

The ultimate goal of this research is to reach a model that would output a valid placement for sizings of different circuit topologies. However, for this to happen the model has to accept different sized inputs, as each circuit can have a different number of devices. In this regard, it would be necessary to use a Recurrent Neural Network (RNN) to accept a variable number of inputs. This would raise the necessity for an increase in the quantity of data used to train the network, as the model will get more complex.

REFERENCES

[1] N. Lourenço, R. Martins, and N. Horta, *Automatic Analog IC Sizing and Optimization Constrained with PVT Corners and Layout Effects*. Springer, 1 ed., 2017.

[2] M. P.-h. Lin, Y.-w. Chang, and C.-m. Hung, "Recent Research Development and New Challenges in Analog Layout Synthesis," *Asia and South Pacific Design Automation Conference*, pp. 617–622, 2016.

[3] R. Martins, N. Lourenço, and N. Horta, "Multi-objective optimization of analog integrated circuit placement hierarchy in absolute coordinates," *Expert Systems with Applications*, vol. 42, no. 23, pp. 9137–9151, 2015.

[4] A. Handkiewicz, S. Szcze¸sny, M. Naumowicz, P. Katarzyński, M. Melosik, P. Śniatała, and M. Kropidłowski, "SI-Studio, a layout generator of current mode circuits," *Expert Systems with Applications*, vol. 42, no. 6, pp. 3205–3218, 2015.

[5] S. Bhattacharya and N. Jangkrajarng, "Multilevel symmetry-constraint generation for retargeting large analog layouts," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 6, pp. 945–960, 2006.

[6] L. Zhang, N. Jangkrajarng, S. Bhattacharya, and C. J. R. Shi, "Parasitic-aware optimization and retargeting of analog layouts: A symbolic-template approach," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 5, pp. 791–802, 2008.

[7] P. H. Wu, M. P. H. Lin, and T. Y. Ho, "Analog layout synthesis with knowledge mining," in *2015 European Conference on Circuit Theory and Design (ECCTD)*, pp. 1–4, 2015.

[8] P. H. Wu, M. P. H. Lin, T. C. Chen, C. F. Yeh, X. Li, and T. Y. Ho, "A Novel Analog Physical Synthesis Methodology Integrating Existent Design Expertise," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 2, pp. 199–212, 2015.

[9] P. Domingos, *The Master Algorithm: How the Quest for the Ultimate Learning Machine Will Remake Our World*. New York: Basic Books, 2015.

[10] 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, 2002.

[11] D. B. Fogel, "The Advantages of Evolutionary Computation," in *Biocomputing and Emergent Computation: Proceedings of BCEC97*, pp. 1–11, World Scientific Press, 1997.

[12] G. Liou, S. Wang, Y. Su, and M. P. Lin, "Classifying Analog and Digital Circuits with Machine Learning Techniques Toward Mixed-Signal Design Automation," in *2018 15th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*, pp. 173–176, 2018.

[13] R. M. F. Martins, N. M. T. Lourenço, A. Canelas, and N. Horta, "Stochastic-based placement template generator for analog IC layout-aware synthesis," *Integration*, vol. 58, pp. 485–495, 2017.