1.4 Machine Learning Overview

Since the work have a huge component of machine learning, it is useful to give an overview of the used methods in order to a better comprehension of the models proposed in the state-of-the-art. It is also important to review some dimensionality reduction methods as well as model selection methods, which are important to tune the model with the best hyperparameters.

1.4.1 Supervised and Unsupervised Learning

ML systems can be divided into four classes: supervised learning , unsupervised learning, semi-supervised learning, and reinforcement learning.

Supervised Learning

This approach means having a full set of labeled data while training an algorithm. For each example in the training dataset is tagged with the answer the algorithm should come up with. This training approach makes the model capable to, when receive new data, compare it with the training data and make a prediction. In the Figure 1 is presented a flow diagram of the supervised learning approach.

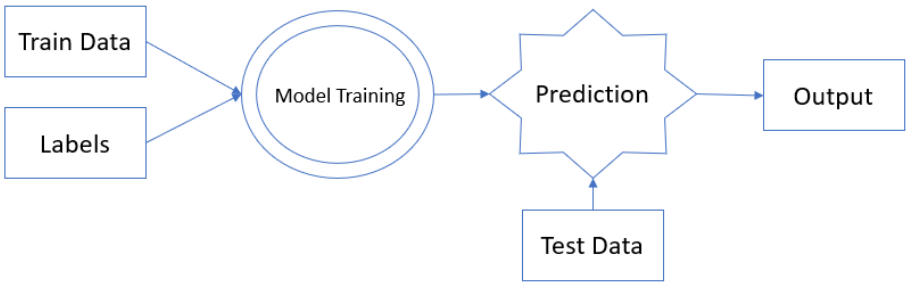

Figure 1 – Supervised Learning Flow

In this type of learning is where all the machine learning methods can be applied, such as Linear Regression, Support Vector Machine, Logistic Regression, Naïve Bayes among others since those methods required labeled training data to be able to compare a new data with it in order to provide an accurate prediction.

Unsupervised Learning

As the opposite of supervised learning, this approach has no label, and its goal is to describe the association, relation, and patterns among the set of input measures. This is an growing area since obtaining labeled datasets are not easy. This type of learning can be applied using neural networks, which attempts to automatically find relationship in the data by extracting useful features and analyzing its structure. Accordingly, to the problem, the unsupervised learning can organize the dataset in the following ways:

1) *Clustering:* The deep learning model looks for training data that are similar to each other and group them together. After that a several of clustering methods can be applied such as, for example, C-Means Clustering or Hierarchical Clustering

   Anomaly detection: In this approach the deep learning model can detect the outliers in a dataset.

2) *Association:* The deep learning model can predict the remain attributes by looking at a key attribute on a certain data point.

3) *Auto Encoders:* This approach takes the input data, compress it into a code and tries to recreate the unput data from that compressed code. In images this approach can remove visual noise improving the image quality. Another application of this method is the fact that looking to the given input data it can randomly generates new data that is similar to the input one. This neural network is also commonly used for dimensionality reduction.

Semi-Supervised Learning

It is the intermediate learning process, where in the training set some data have labels and other do not.

This method is useful when labeling data is time-consuming and very difficult, for example medical images.

For this purpose general adversarial networks (GANs) are commonly used, this network is divided in two networks (as it can be seen in Figure 2), the first one is called generator and is responsible for create new data that is similar to the input one, the second one, which is called Discriminator, is responsible do evaluate the new inputs and decide if they make part of the training set or not [21].

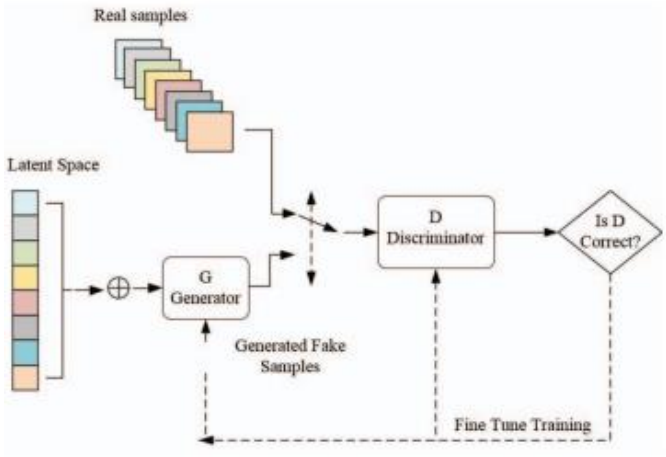

Figure 2 - GANs Design Flow (from [21])

Reinforcement Learning

Reinforcement Learning (RL) also stands between supervised and unsupervised learning, because no set of action is given to the agent, and it acts over an environment where sequential decisions need to

be made with limited feedback. The main goal is to develop an agent capable to interact with environment and learn how to behave on it and depending on the actions taken it receives rewards or punishments which are the only information the agent has, usually this is try-error-try way [22]. The agent objective is to develop a policy that indicates which action should be taken in order to maximize the reward. To attain that goal, the agent needs to explore and interpreted the given reward and come up with optimal decisions for any state. One relevant characteristic of RL is that in any situation the agent needs to choose between exploiting its knowledge or explore actions that never tried before. To clarify, the Figure 3 shows the reinforcement learning design flow previously explained.

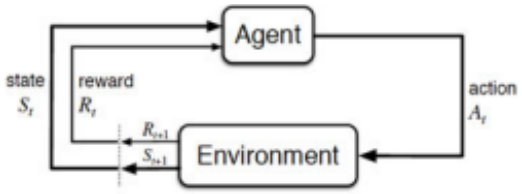

Figure 3 - Reinforcement Learning Design Flow (from [22])

1.4.2 Tribes of Machine Learning

Machine Learning is a vast area with a vast number of methods to be applied, and in [22] the authors define five tribes in which ML area can be divided, illustrated in ?. All the five tribes have unique methodologies so, when using ML, is important to know the methods specifications and analyze the different approaches and then select the one that better adapts to the problem.

Table 1 - Tribes of Machine Learning (from [22])

Tribe	Origins	Master Algorithm	Examples of Methods
Symbolist	Logic	Inverse Deduction	Decision Trees
Connectionist	Neuroscience	Backpropagation	Deep Neural Networks, Perceptron
Evolutionaries	Evolutionary Biology	Genetic Programming	Genetic Algorithm
Bayesians	Statistics	Probabilistic Inference	Naïve-Bayes, LDA, PCA
Analogizers	Psychology	Kernel Machines	SVM, KNN

Briefly, the Symbolist tribe use symbols, rules, and logic to represent the knowledge. The Connectionist tribe aims to recognize relationships and patterns between inputs and outputs in a dynamic way using weights, which are adjustable. The Evolutionaries tribe generates population and obtain the fitness of that population and decide who will reproduce, or not, in order to generate offspring's which are better than the previous population in order to achieve the goal through generations. The Bayesians tribe is based on the bayes theorem to obtain the likelihood value. Finally, the Analogizers tribe aims to optimize a function taking in consideration the constraints.

In light of this work, modulate the behavior of a circuit or device, obtaining for example the output waveforms, is a regression problem. The Bayesians tribe and Analogizers tribe are commonly used (but not only) for classification problems, so they are discarded from the possible methods to use in this

work. The Symbolist tribe is also discarded because in cases with a lot of constraints developing an efficient solution presents a tremendous challenge and is time-consuming. As mentioned in the motivation chapter of this work, one of the main goals when applying Machine Learning to analog circuits is to improve the simulation time, so Evolutionaries tribe approaches might be discarded due to the simulation time that is associated with natural selection of the most fit individuals in each different strategy.

As mentioned in previous sections, the use of ANN for modulating circuits behavior have been widely researched and discussed. This neural networks with the number of hidden layers, number of neurons and weights properly set are capable to provide accurate results with very fast simulations. Considering all the tribes and the reasons provided, the ANN methodology is the chosen one for this work.

For that purpose, a brief explanation of the Artificial Neural Networks methodology is presented in the following section.

1.5 Artificial Neural Network Overview

Widely researched, ANNs prove to be flexible, adaptative to any type of problem, from the simplest to the more complex ones, and performed predictions in a short time of simulation. Those advantages allying to the fact that ANN are very efficient, is the main reason why it is used in this project, because it is pretended to have an efficient modulation of the circuit behavior in a short period of time.

In this section the ANNs general architecture and behavior are explained as well as its hyperparameters, their influence on the model behavior and how to automate its choice. Since ANNs required data well preprocessed to train the model it is important to explain some essential feature engineering and engineering data analysis.

1.5.1 General Architecture

First is important to define the basic unit of the ANNs, the Neuron, which is the elementary artificial neural network capable of being trained. The artificial neural network is a composition of numerous neurons. In Figure 4 is represented a Neuron.

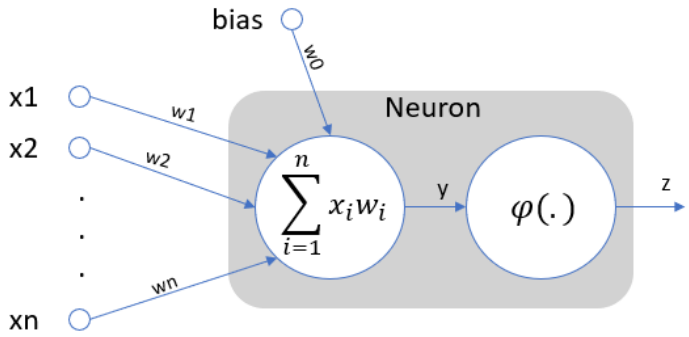

Figure 4 – Elementary ANN - Neuron

In the equation (1) is given the output value of a neuron:

$$z = \phi\left(\sum_{i=0}^{n}(w_i x_i) + b w_0\right) \quad (1)$$

The x_i are the input values, ϕ is the activation function that is explained in section 2.3.3, w_i are the weights and b are the bias, which are hyperparameters and its tuning is explained in section 2.3.5. If bias is considered to be 1, the previous equation can be rewritten is a simplistic way presented in the equation (2).

$$z = \phi\left(\sum_{i=0}^{n}(w_i x_i)\right) \quad (2)$$

To deal with more complex problems, the neuron itself is not capable to capture patterns and behaviors between inputs and outputs of non-linear problems, so neurons linked to the others originate an multilayer perceptron, which is capable to deal with more complex problems, the ANN is illustrated in Figure 5.

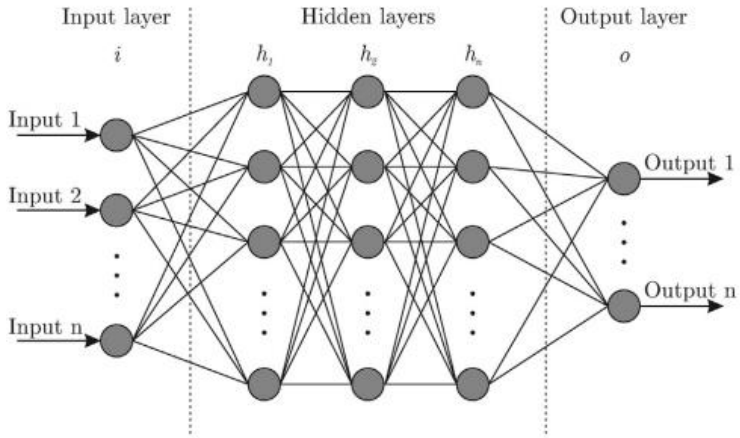

Figure 5 - ANNs General Model (from [24])

ANNs have the input layer with number of neurons equal to the number of features present in the dataset, the output layer has the same number of neurons as the number of outputs desired. The number of hidden layers and its number of neurons is a hyperparameter to be tuned and there is no rule to follow, its tuning is based on the experience of the operator, in section 2.3.5 is given some approaches to decide the best combination of them.

For the case of fully connected ANNs all the neurons are linked to each other's, meaning that the input of the next layer neurons are the outputs of all the previous layer neurons as Figure 5 is showing. The input layer receives the sum of the input values multiplied by their respective weights and apply an activation function generating the output of the neuron in the hidden layers or in the output one, the output of the previous layer neurons is multiplied by their respective weight and then sums them all, and the result pass through an activation function which gives the output of the neuron. This learning process consists of adjusting the neural network weights so that the network's transformation of input into outputs optimizes the loss function [24].

The main objective of the model's designer is to create and train a model, with limited examples as references, in order to obtain an efficient model capable to map a given an input generate to an output that is close to the expected one. There are enumerous types and variants of the ANNs, despite being easy to interpret the output value is hard to materialize and describe the behavior of the network during the training and test phase.

As the complexity of the problem increase new types of ANNs are developed, such as recurrent neural networks, Elman recurrent neural networks, long-short term memory recurrent neural network, but all of these variants keep the same principle of the multilayer perceptron explained previously.

1.5.2 Backpropagation

The learning phase of the ANN model consists of updating the weights of each neuron using the backpropagation algorithm. Given the inputs of the network and the initial weights and bias, the output ($\hat{y}$) is obtained by forward propagation, as explained in 2.3.1. The output ($\hat{y}$) is compared to the desires output (y) computing the loss L between them. Since this work deals with regression problems the appropriate loss function is the Mean Squared Error (MSE) given by (3).

$$L = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{3}$$

The derivative of the loss with respect to the weight is compute for each neuron's weight and consequently the weights are updated according with the equation (4).

$$w_{i+1} = w_i - \Delta w \tag{4}$$

The w_{i+1} is the new value of the weight, w_i the previous weight value and Δw the wight update proportion, which is given by the equation (5).

$$\Delta w = \left(\eta \frac{\partial L}{\partial w} w_i\right) \tag{5}$$

The η is the learning step which controls how quickly or slowly the neural network learns a problem. A large learning step makes the model to learn faster at the cost of arriving to a sub-optimal solution for the weights, and a small learning step makes the model learn slow but reach an optimal solution or even the globally optimal solution for the weight's values, so the value of this parameter should be chosen with criteria and its tuning is explained in the section 2.3.5.

1.5.3 Optimizers

To optimize the weights value exists a numerous type of algorithms to perform that task. The Gradient Descent is the most used algorithm and backpropagation make use of it to train the neural network. It calculates in which direction the weights should be altered so that the loss function can be minimized. If we compile the equation (4) with (5) is obtained the Gradient Descent equation in (6).

$$w_{i+1} = w_i - \left(\eta \frac{\partial L}{\partial w} w_i\right) \quad (6)$$

For relatively simple problems this optimizer is a well-chosen approach, although in more complex problems it might take very long time to converge and might even be stuck at a local minimum. To fight those problems is useful to use variants of this optimizer, the most common on is the addition of a Momentum Term to the Gradient Descent optimizer, it accelerates the convergence making the weights flow in the relevant direction reducing the fluctuations in the irrelevant direction. Although, as it is represented in the equation (7), it requires the tunning of one more parameter which needs to be carefully chosen.

$$w_{i+1} = \boldsymbol{\gamma}.\boldsymbol{w_i} - \left(\eta \frac{\partial L}{\partial w} w_i\right) \quad (7)$$

If the momentum term coefficient ($\boldsymbol{\gamma}$) is to high it may skip a local minima and keeps increasing indefinitely so making to the Gradient Descent optimizer, already provided with the Momentum Term, a slight alteration to modify the current weight value (called Nesterov Accelerated Gradient in equation (8)) is possible to slow the algorithm when approaching a local minima and them accelerate it in the other occasions solving the momentum term problem.

$$w_{i+1} = \boldsymbol{\gamma}.\boldsymbol{w_i} - \left(\eta \frac{\partial L}{\partial w} (w_i - \boldsymbol{\gamma}.\boldsymbol{w_i})\right) \quad (8)$$

1.5.4 Regularization

Machine Learning models, including ANNs, are subject to overfitting. A model is said to have overfitting when the error between the proposed curve referred and the data is zero or almost zero, so as the proposed curve fits in perfection the data curve, leading to a model that is too well adjusted to the given training data. This effect leads to a model that is not generalized [25] leading to a negative impact on the performance of the model in unseen data in Figure 6 is possible to see the overfitting effect, where the testing error (when model performs in unseen data) starts to increase significantly.

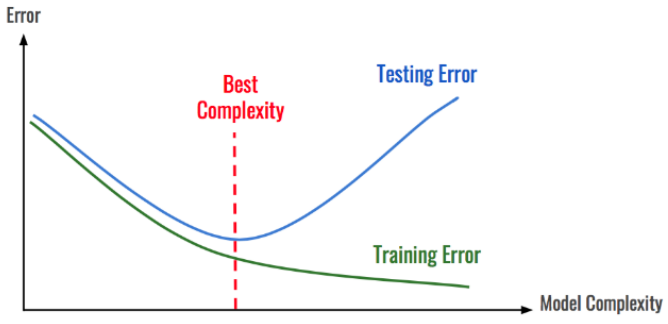

Figure 6 – Overfitting (from [26])

To prevent this problem is advisable to simplify the model, not using high number of neurons in the hidden layers as well as not use a large number of hidden layers, this parameters tuning is explained in section 2.3.5. Another criteria and methods can be applied to deal with overfitting such as introducing a stop criteria called Early Stopping, that means if in a predefined number of iterations the testing error

keeps increasing the train will stop to prevent the overfitting, if we look to the Figure 6 the “Best Complexity” point is exactly where the Early Stopping method should stop the training process, it is the point where after that if the model keeps training it will lead to the overfit effect. Another approach is to add a penalty (L1 or L2) to the loss function so that the model is obligated to make compromises on the weight’s values since they can no longer be arbitrarily large, or it will penalize the loss function. This generalizes the model and helps to prevent the overfitting. Another extremely effective approach is to add dropout layers and dropout connections, this deep learning techniques allows to randomly deactivate neurons and connections during the training phase making the network redundant since it can no longer rely on the information in specific neurons and connection because they might be deactivated making the model more general. Posterior of the training phase all the neurons and all the connections are reestablished [26].

1.5.5 Hyperparameters Tunning

The main disadvantage of the ANN is the enormous amount of hyperparameters that should be adjusted in order for the model perform as well as expected. In this sub-section is provided an overview of each hyperparameter, explaining not only their importance in the network but also strategies to find the most appropriate values.

Hidden Layers

The more hidden layers the more complex is the ANN. A complex neural network is not always synonymous of good results, because the model tends to have overfitting effect in the training set. Apart from this disadvantage, a complex network increases the time and computational power required in the training phase. In contrast a ANN with a low number of hidden layers might not be able to capture the input-output relationship accurately. So is important to pay attention when choosing this parameters values, unfortunately there is no formula or metric to choose the correct number of hidden layers. The current approaches are based on experience, intuition, and slight adjustments, which consists in gradually increase the number of hidden layers until the results (i.e., accuracy) show no further improvement, meaning that adding more layers can increase the overfitting effect.

Hidden Neurons

Since the number of neurons in the input layer is determined by the number of features and the number of neurons in the output layer is determined by the number of desired outputs, only the number of neurons in the hidden layers need to be carefully chosen, since a large number of it may lead to overfitting effect and a small one may lead to underfitting effect. Once again there is no formula or metric capable to define the right number of neurons in that layer. So, the choice of the number of neurons relies on the model’s designer experience and in some heuristic methods present in [27] and in [28] where the number of the neurons in the hidden layer should be between the number of input layer neurons and the number of output layer neurons, the hidden layer size should not have more neurons than twice the number of neurons in the input layer. However, as mentioned buy the author of [28], despite this heuristic approaches the number of hidden neurons always depends on the problem. Apart from this heuristic, try-error is always a good approach such as start with the same number of

hidden neurons for every hidden layer and gradually increase the number of neurons, taking in consideration the performance, until the point where the model starts overfitting.

Learning Rate

This parameter is important when the ANN is updating the weigths. This hyperparameter indicates how fast the learning is performed, in other words, how fast the model responds to the estimated error in each update and the capability to converge to a minimum. As mentioned in section 2.3.2 choosing this parameter is a hard task since a large learning rate makes the model to learn faster at the cost of arriving to a sub-optimal solution for the weights, and a small learning rate makes the model learn slow but reach an optimal solution or even the globally optimal solution for the weight's values. As any other hyperparameters there is no function or metric to pick up the right value, so try-error is the only way to do that. So since is desired a model that performs faster, start choosing a high value for the learning rate and gradually decreasing it until the model performance not improve can be an approach to find the correct learning rate for each problem.

Activation Function

According to the neurons value, the activation function decides if the neuron should be activated or not and its behavior and capabilities. There are an enormous variety of activation functions, that already were studied, which have their purpose, advantages, and disadvantages. In backpropagation algorithm is important to have differentiable activation functions, otherwise this algorithm does not work since it is not able to compute the loss derivative in respect to the weigths.

In regression problems, where the output is a real number, the activation function should be linear (Figure 7) or a Rectified Linear unity (ReLu) computed through $f(z) = \max\{0, z\}$ (Figure 8). The main advantage of the ReLu is being a non-linear function capable of encode non-linear relationships. The main disadvantages are since it does not have upper bound it is susceptible to output values very high comparing to the desirable ones and in z = 0 there is a discontinuity that may lead to an optimization with more errors that desired.

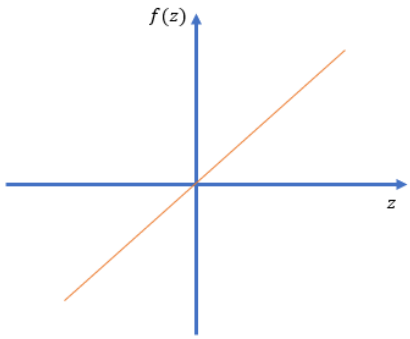

Figure 7 - Linear Activation Function

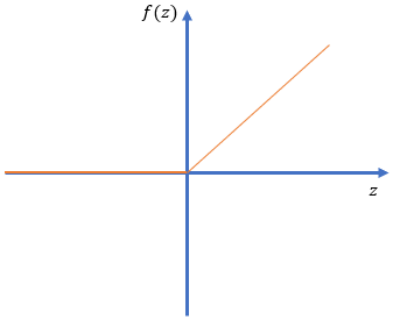

Figure 8 - ReLu Activation Function

To correct the ReLu drawbacks some variations of that method are performed such as Leaky ReLu and exponential linear unity (ELU) function. The ELU in z = 0 is smoothed so it makes the discontinuity problem vanish.

In classification problems if it is being performed a binary classification, the sigmoid or hyperbolic tangent functions are commonly chosen since in the cases where values are too high it establishes a limit. If the problem has multiple classes, it is important to know if it has non-exclusive classes or a mutually exclusive classis. In the first case, one data point can have multiple labels assigned so, the sigmoid function must be used as an activation function. For the second case, if a data point has a certain label, it cannot have another one and in that case is used softmax as activation function.

1.5.6 Feature Engineering and Engineering Data Analysis

In ANNs raw data might be a problem in the training phase. Excessive number of the features can make the model perform worst, so apply the Principal Component Analysis or the Linear Discriminant Analysis can be a way to take the numerous features and extract their relationships creating a dataset with less features. After that, another important issue to deal is the null values which make hard for the ANNs to learn. There are two ways to deal with it, the first one drops all the lines with null values, the second one tries to fill those missing values for example with the mean value of that feature.

Once all data is cleaned it is time for split the dataset into train, validation, and test set. If the dataset is a time series, where the output depends on previous time events, it is important to keep the dataset order when splitting the data. Having all the three datasets, if the data is unbalanced, in order for the model learn correctly only the training data should be balanced keeping the validation and test data unbalanced to mirror the reality world when validating and testing the model. The step before the data entre in the ANN is the normalization phase where all the input values will be normalized into a range between 0 and 1, using either Min-max normalization or z-score. This normalization allows the errors in the network being in the same order of magnitude.