POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Parametric machines: generalization and explainability in deep learning

Tesi di Laurea Magistrale in
Mathematical Engineering - Ingegneria Matematica

Martina Garavaglia, 967257

Advisor:
Prof. Piercesare Secchi

Co-advisors:
Dr. Mattia G. Bergomi
Dr. Pietro Vertechi

Academic year:
2022-2023

Abstract: The lack of a formal definition makes the design, implementation, and deployment of deep neural network a time-consuming and highly specialized task. We test the parametric machine framework—a formal generalization of deep neural architectures—on two classical deep-learning applications: time series forecast and classification. First, we show how novel architectures drawn form the space of parametric machines can compete and perform better than their classical counterpart on an electrocardiogram classification task. There, we introduce a regularization technique for parametric machines and an explainability algorithm allowing us to compute a notion of uncertainty on an input-by-input basis. Second, we employ parametric machines to forecast electrical energy consumption. We compare the performance attained by the machines with comparable classical deep neural networks. Then, we investigate the generalization capabilities of the models.

Key-words: Parametric machines, time series, explainability, sensitivity, generalization

1. Introduction

Deep learning is becoming progressively more challenging to navigate and comprehend: deep neural architectures are increasing in complexity to deal with novel and more complex problems. These architectures are generally hand-crafted and devised following a trial-and-error protocol. Based on highly parallel, nonlinear computations and lacking a natural theoretical framework, deep learning models turn out to be often unintelligible to both

experienced and naive users. Thus, currently, the design and implementation of neural networks requires the intervention of specialized programmers and data scientists making the adoption of such models in real-world scenarios a time-consuming activity. Furthermore, deep neural networks require significant amounts of high-quality data to train and are endowed with only partial generalization abilities. Finally, complex, deep architectures incur in pathologies that are not simply explained by overfitting [4], e.g., vanishing gradient and instability during training.

Attaining a formal definition of neural architecture and thus being able to represent deep learning models as points in a well-defined mathematical space would accelerate the design and implementation of neural networks, make models more easily shareable, and possibly provide guarantees of convergence and stability of such models. We want to showcase through a variety of applications how *parametric machines* [32]—a unified mathematical framework generalizing classical deep-learning architectures—can lead to more efficient model design, interpretable results, and enhance the generalization abilities of classical deep learning models.

Aim We aim to compare classical deep-learning networks and novel models drawn from the parametric machines framework on tasks such as classification and forecast of time-varying signals. We investigate the foundational parametric-machine framework to discuss the generalization power (i.e., the ability to forecast or classify data with different features than the ones belonging to the training set) of parametric machines and probe their internal dynamics to provide a notion of explainability for such models.

Contributions First, we briefly discuss the mathematical foundation of parametric machines. Then, we showcase an application of parametric machines to the classification of electrocardiograms. After discussing the performance of several parametric-machine architectures and comparing their performance with the current state of the art, we implement and discuss an explainability pipeline for parametric machines. We show how such pipeline can be leveraged to provide a measure of the machine uncertainty in classifying test samples, but also yield intuitive visualization accessible to inexperienced users. Another application, which also considers time series, involves studying the energy demand in a particular region of Ecuador. There, we compare parametric machines' performance to other deep learning architectures, and test the ability of both classes of models to produce reliable forecast on test signals, endowed with different statistical properties than the training samples. In both cases, it is equally crucial to maintain a practical perspective towards the specific objectives set by the case studies, and consequently ensure that they could be beneficial to the end-user.

Structure In section 2 we introduce the fundamental notions of parametric machines covered extensively in [32]. Section 3 focuses on the first practical case of time series classification using clinical data. The section introduces the architectures and presents the best results achieved. In section 4, the concept of explainability is explored through sensitivity maps obtained from the parametric machine architecture used on these data. Then, we use dimensionality reduction and clustering algorithms to provide practical advice to potential users of this methodology. Section 5 considers a second practical case involving time series forecasting of energy consumption data. This section also discusses the architectures used and showcases the results obtained. In section 6, we explore the generalization property of parametric machines (i.e., the ability to generalize on users with significantly different distributions).

2. Theoretical framework

Using a well-defined mathematical framework in which differentiability is guaranteed allows us to use optimization techniques avoiding pathologies, hence backpropagation (computation of loss function's gradient at each step updating weights to minimize loss) is ensured by machines mathematical properties. The intuition behind machines is that neural networks can be considered as an endofunction $f : X \to X$ on a space of global functions X (defined on all neurons on all layers). Instead, in a classical deep learning framework, different layers in a network are combined using composition. However, this framework brings with it some disadvantages: shortcut connections are not supported and non-sequential architectures fail to be created.

In [32] the authors consider a global space $X = \bigoplus_{i=0}^{d} X_i$ and the global endofunction

$$f = \sum_{i=1}^{d} l_i \in C^1(X, X)$$

In order to establish a relation between the composition of functions and the sum of functions, the output space of the network is considered to be the entire X and not only the last layer space X_d. Thanks to the sum-based structure of the global function, layers are no longer required to be sequential, but they must obey a weaker condition of independence (see [32, Sec. 2.3]).

In particular, by composing independent machines of depth one it is possible to devise architectures of arbitrary complexity. This allows us to deal with complex, high-dimensional data as in the classical neural network framework. For a formal description of depth in the context of parametric machines, we refer the reader to [32, Sec. 2.2].

Architectures Shortcut connections emerged in the recent literature [8, 15, 21] to overcome pathologies such as the vanishing gradient problem and the degradation problem. The theoretical framework introduced so far provides naturally a definition of *complete* architecture—i.e., an architecture with all shortcuts [32, Sec. 2.3]. This rich connectivity among layers augments the computational cost of the model. However, this issue is tamed by the framework itself.

Moreover, the mathematical framework introduced in [32] creates a searchable space of architectures. It is also natural to seek special cases which generalize existing architectures. In particular, we shall recall here the definition of dense, convolutional, and time machine. The implementation of these examples is available in [5]. Here, we provide an intuitive definition and exemplification of these notable machines. See [32, Sec. 3] for details.

Firstly, we discuss feedforward architectures (dense and convolutional machines), in which the information moves in only one direction. As we can visualize in fig. 2, panel a), all layers of higher depth take knowledge from all layers of shallower depth, building a network with complete shortcuts.

A different type of architecture is the time machine. Here, an additional knowledge dimension is added: the timestamp. All layers, as well as learning from the previous layers, also evolve with past knowledge, as we can see in fig. 2, panel b). In practice, a time machine is a hybrid of recurrent and convolutional architectures, based on parameters choice.

3. Classification

Affordable hardware and effective data compression means acquiring vast amount of high-resolution data is becoming common practice in modern healthcare pipelines. While data acquisition is gradually becoming effortless, massive, highly-variable data sets are still hard to process. However, deep learning methods are particularly effective when trained on large, high-quality data sets. In particular, such techniques showed tremendous potential in healthcare applications and have been successfully applied to a wide variety of tasks [24], e.g., medical image analysis [30], disease diagnosis and prediction [23], drug discovery [19], and personalized medicine [26]. Moreover, the nonlinear nature of deep learning algorithms allows them to detect patterns and correlations that may be difficult for humans or traditional machine-learning algorithms to identify (see, e.g., [27]), and perform with high accuracy in extremely specialized diagnostic tasks [11].

Despite the promising results, there are still challenges to be addressed in the usage of deep learning in healthcare, such as ensuring the reliability and explainability of the models' output, guaranteeing robustness to noisy inputs, and overcoming regulatory and ethical issues.

3.1. Dataset

Aim and motivation We aim to test parametric machines on time series classification and provide strategies to regularize and interpret the parameters learned by the machine during training. With this aim in mind, we consider the ECG200 dataset [34], a benchmark data set for time series classification. Each series traces the electrical activity recorded during one heartbeat. Time series are labelled as normal or abnormal heartbeats (myocardial ischemia). Importantly, alterations of the hearbeat signal due to ischemia can be extremely varied. This variability and the complexity of the mechanics underlying heart dynamics make the ECG200 dataset suitable for testing novel techniques and architectures such as parametric machines.

Medical setting Myocardial ischemia occurs when blood flow to the heart is reduced, leading to necrosis of heart muscle due to lack of enough oxygen. The reduced blood flow is usually the result of a partial or complete blockage of the heart's arteries [28]. This pathology can be recognized observing different locations of electrical signals in the heartbeat, namely the ST and T-wave segments. The ST segment is the plateau phase where the potential differences in the heart remains relatively steady. In healthy patients, this phase has a long duration which enables the majority of the ventricular myocardium to contract simultaneously. The T-wave is instead representative of the rapid repolarization phase (i.e., potential decrease) and can have a negative or positive slope. The ST and T-wave locations depend on each other. Indeed, changes in the ST segment are typically followed by changes in T-wave. Ischemia affects the plateau phase in which the ST segment may be depressed (or elevated) respect to a normal heartbeat. The T-wave may decrease in amplitude, become negative or increase markedly (see fig. 1). The changes in the signal depend on the localization, extension and timing of the ischemia. It is important to note that the ECG is only a preliminary diagnostic tool used to determine whether there may be evidence of ischemia. It is not typically considered to be a definitive diagnostic test, and further evaluations and assessments may be necessary to make a final diagnosis.

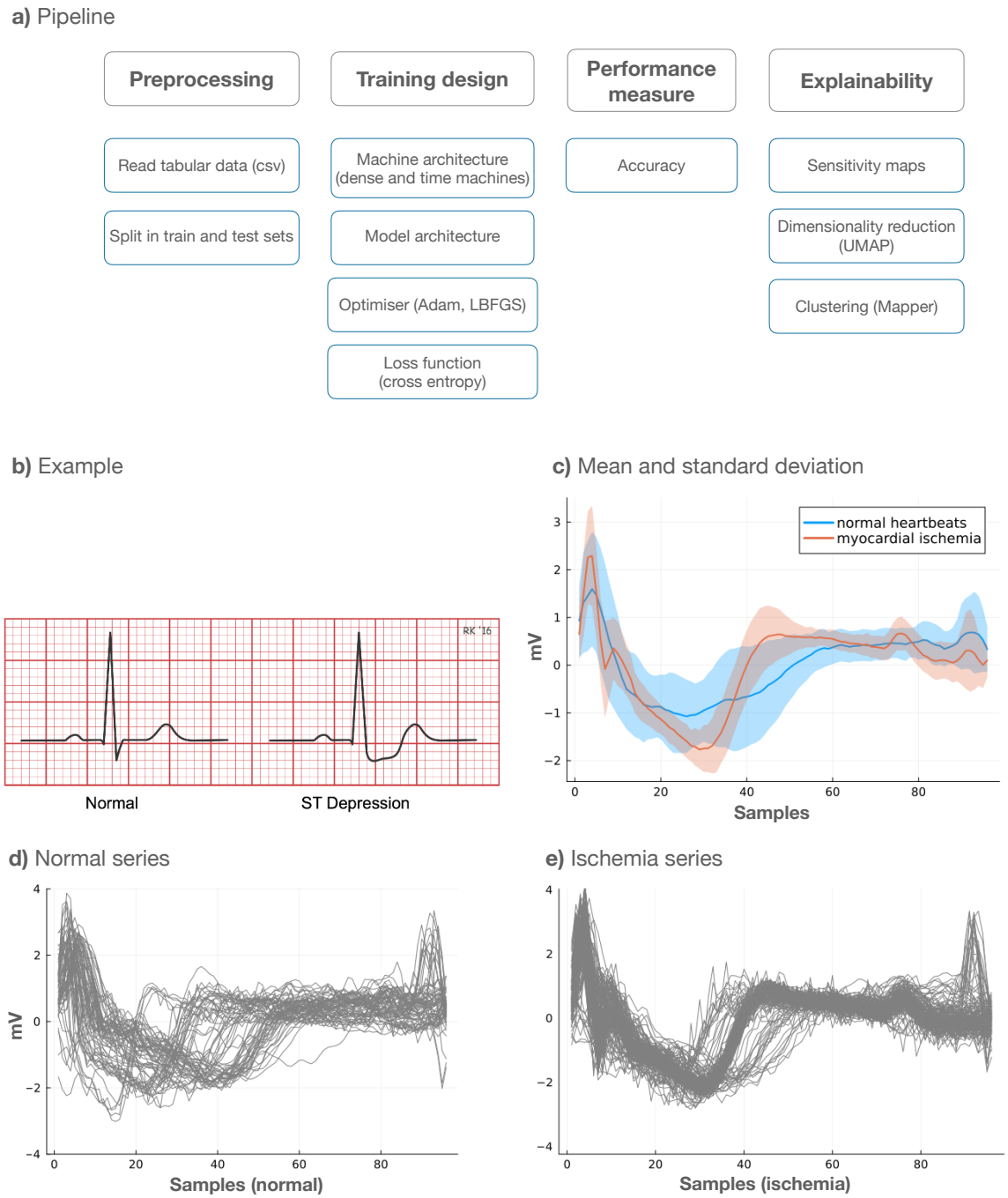

Figure 1: The ECG200 dataset is composed of 200 heartbeat samples labelled as normal or abnormal. Each time series consists of 96 electrical potential measurements (mV). **a)** Work pipeline. **b)** Example image of a normal heartbeat and a myocardial ischemia heartbeat. Adapted from [18]. **c)** Mean and standard deviation of each sample of ECG200 split by label. **d)** Normal heartbeat time series. **e)** Abnormal heartbeat time series.

Data structure The ECG200 dataset consists of predetermined train and test set. Both sets are composed by 100 time series, each comprised of 96 observations. The difference in ECG between a normal heartbeat and a myocardial ischemia can be better seen by analyzing the average of all measurements. See fig. 1 for an encompassing representation of the ECG200 dataset.

3.2. Methods

In the following paragraphs, we discuss the preprocessing, model selection, and training pipelines we devise to analyze the ECG200 dataset.

Preprocessing The ECG200 dataset comes already split into training and test sets. Each set contains 100 samples of either normal or abnormal heartbeats. The ECG200 samples are bounded and do not present outliers. For this reason, we designed an extremely simple preprocessing pipeline encoding labels as one-hot vectors and adding a channel dimension to the data, as it is customary in deep-learning practice.

Architectures As mentioned above, we consider two types of architectures: the dense and time machines. In the dense machine case, we choose the sigmoid function as nonlinearity, which constrains output values between zero and one. The model also includes a dense output layer. For time machine, we divide the global space into six subspaces, each of size 16. Moreover, we use the sigmoid function as nonlinearity and a timeblock of length 16. The model also includes a convolutional output layer.

Training and testing We set the optimizer as well as the loss function we want to work with. During this stage, we explore several possibilities. We finally chose cross entropy as the loss function and we found that the ADAM optimizer is the most effective option for this purpose due to its simplicity with a learning rate of 0.005.

During training, we save the best model parameters computed using backpropagation and performance measures on every training epoch. Finally, we visualize performance measures fig. 2 to gain insight into how well the model performed using accuracy measure.

3.3. Regularization

Regularization is a technique commonly used in deep learning to prevent overfitting. Overfitting occurs when a model is too complex and overspecializes its parameters to perfectly fit the training data. This specialization results in poor generalization to data endowed with different features or following a different distribution than the training ones. Regularization involves adding a penalty term to the loss function of the neural network during training. As an example, the penalty term can be proportional to the squared magnitude of the weights in the network, which encourages the network to learn smaller weights and thus learn smoother solutions.

In our scenario, the regularization term is more sophisticated. The smoothing process is only applied to the temporal dimension, which involves squaring the difference between model weights in two consecutive data points in the time series:

$$\tau = \lambda \cdot \sum_{t=1}^{T} (w(t) - w(t-1))^2 \tag{1}$$

When performing regularization on the time dimension, the goal is to encourage the model to learn smooth patterns over time. By regularizing only on the time dimension, the model is encouraged to learn patterns that are consistent over time, which can improve its ability to predict future outcomes. Additionally, this approach can help to reduce the impact of outliers or noise in the data that may be present in individual time steps, but not across the entire time series. In practice, the smoother the weights in time, the smaller the regularization term will be. The strength of time smoothness regularization is controlled by a hyperparameter λ, which determines the tradeoff between fitting the training data well and keeping the weights smooth. Larger values of the regularization parameter result in more emphasis on the penalty term, and therefore smaller weights in the network.

3.4. Results

Using dense machines, the training process is monitored through the value of the loss function on the train set and of the accuracy both on the train and on the test set (see fig. 2).

The results surpass the state of the art achieved by other models with an accuracy of 0.9 (for more details, see [34]).

A dense machine accurately represents the problem due to its simplicity. With time machine, the achieved accuracy is 0.91, and once again, above the state of the art. The training loss and accuracy can be found in fig. 2.

Moreover, in fig. 2, we can observe dense and time machine losses after adding a regularization parameter of 0.01. In this case we can observe a dense model accuracy of 0.91, higher than that of the state of the art. Regarding the time machine, accuracy is equal to 0.9 with a regularization parameter of 0.01.

We have also tested a convolutional machine's architecture, but despite obtaining quite satisfactory results, we decided not to go in depth here as the characteristics of this network are not exploited on this type of data. If the reader is interested in these results, they can consult [12].

4. Explainability

The word explainability refers to the ability to understand and interpret the decision-making process of a machine-learning model in an input-dependent fashion. In many applications, such as healthcare [31] and finance [33], explainability plays a crucial role: a medical doctor, as well as an auditor, needs access to the internal mechanisms dictating the model's output. There are techniques that can be used to increase model explainability, and several methods have been developed specifically for deep learning models. For example, layer-wise relevance propagation is a technique for attributing importance scores to individual neurons in a deep neural network [7]. Integrated gradients is another technique that can be used to attribute importance to individual features in a deep learning model [29]. Finally, there is ongoing research in the field of explainable artificial intelligence (XAI) to develop more comprehensive and standardized methods for model explainability [9]. The goal of XAI is to enable users to understand and trust the decisions made by machine learning models, especially in sensitive applications where the consequences of errors could be significant.

In this sense, our aim is to devise a technique that could effectively communicate the workings of deep learning models to individuals who lack expertise in the field, by utilizing sensitivity maps.

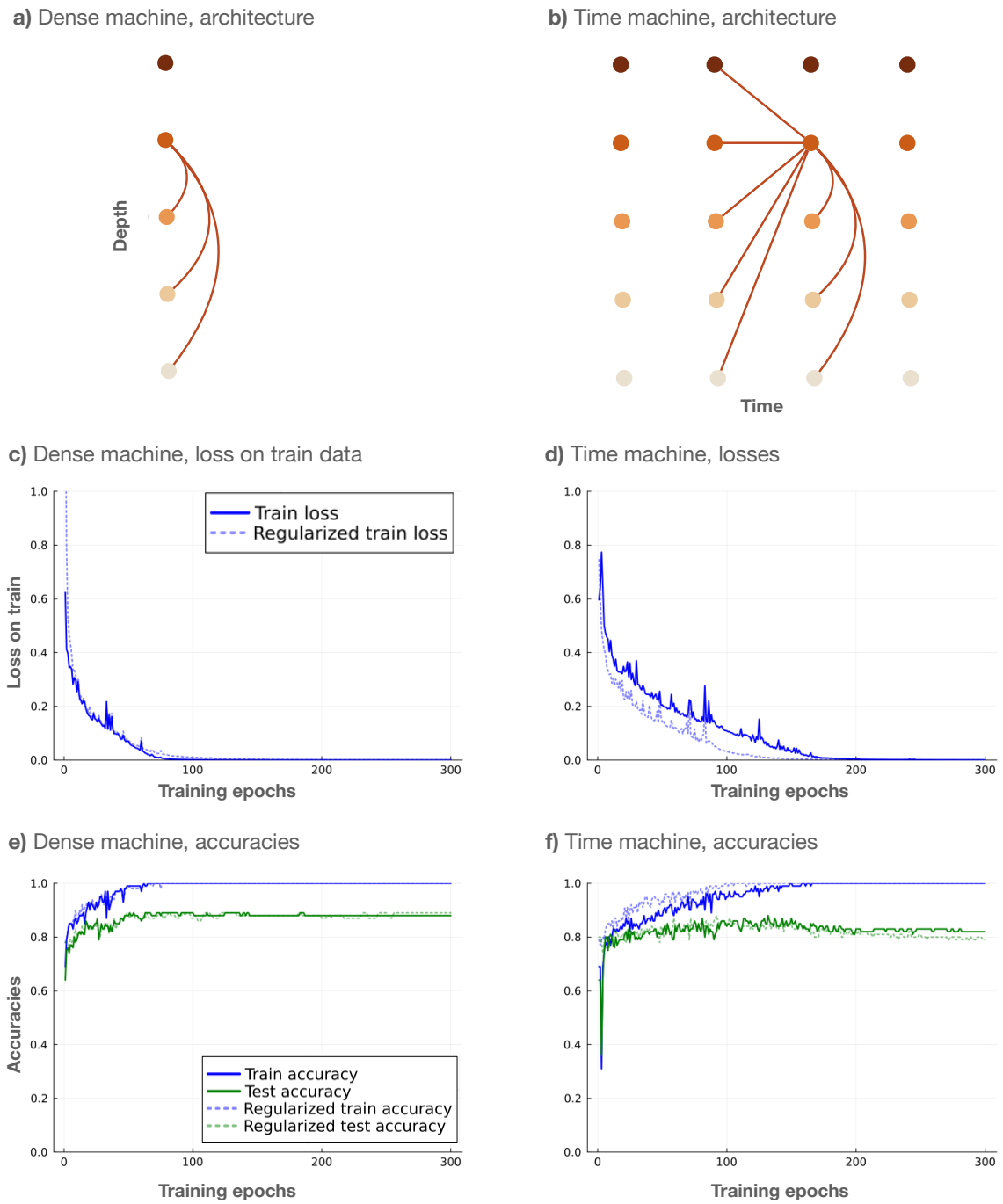

Figure 2: Loss and performance. **a)** Feedforward machine architecture showing the shortcuts structure. **b)** Time machine architecture showing the shortcuts structure. **c)** loss on train data with and without regularization for dense machine during 300 epochs. **d)** accuracy on train and test data with and without regularization for dense machine during 300 epochs. **c)** loss on train data with and without regularization for time machine during 300 epochs. **d)** accuracy on train and test data with and without regularization for time machine during 300 epochs.

4.1. Sensitivity maps

In section 3, we developed a model that provides predictions with 91% accuracy, but it may not be sufficient in a medical context. It is essential for models employed in sensitive contexts to be endowed with a mechanism that can determine the degree of uncertainty of the model and provide guidance to medical doctors, highlighting regions of the input that could potentially invalidate the model's output. Therefore, the aim of this section is to provide a mathematical interpretation of how machines operate on input data, enabling

- detection of critical regions of the input sample;
- extraction of relevant samples from the training set that could help identify the detected criticalities;
- quantification of the model's uncertainty on an input-by-input basis.

Each machine consists of a non-sequential juxtaposition of linear and nonlinear components. We use the sigmoid function to illustrate the construction of explainability maps for parametric machines that we shall call *sensitivity maps*. However, the following construction holds for any pointwise nonlinear function. For the sake of intuition, we can think about the sigmoid function (i.e. the activation function we utilize in section 3) as a piecewise linear function defined on three intervals: first, the function is nearly flat and tends towards zero. Then, the function has a positive slope, and hence positive derivative. Finally, the function returns to be constant and of value one. The input data traverse this function, crossing through these three sections. The data that pass through the outermost sections will not contribute to the output because they have almost zero derivative. Instead, points mapped to regions of positive slope contribute actively to the model's output. Practically, we compute the derivative of the nonlinear function with respect to the machine's output before the nonlinearity is applied to the input. We call this construction a sensitivity map. In symbols, we express the sensitivity ρ as

$$\begin{aligned} y &= W * z + x_0 \\ z &= \sigma(y) \\ \rho &= \sigma'(y) \end{aligned}$$

where y is the machine's output before the nonlinearity σ, W is the weights matrix, x_0 is the input vector and z the machine's output after the nonlinearity.

The implications of this concept can vary depending on the type of machine used to develop our models.

Dense machine The sensitivity map of a dense machine can be used to efficiently compute optimal depth, contrary to the stochastic approach developed in [16]. Figure 3 panel a) shows the sensitivity maps for normal and abnormal data samples. We believe that changes in values of sensitivity on a given dimension correlate with the classification performance of the model.

Time machine In time machines, the sensitivity map assumes a different meaning. In this scenario, the sensitivity map provides insights into individual time series by identifying the specific time points and learning depths where the model is more sensitive to the signal, as we can see in fig. 3, panel b).

Moreover, the mean of the first rows of the sensitivity map (e.g., the channels of the input layer) tells us which part of the signals is more sensitive during training. In time machines, the first layer holds the most significant representation of the inputs. Therefore,

incorporating this information when analyzing time series data can offer us a perspective on how to interpret the sensitivity map. It is important to note that this is just one of the many possible interpretations, and several others will be taken into consideration. In fig. 3, panel c), we can observe that the most sensitive parts of the signal are the relative maxima and the initial slope (e.g., the beginning of the ST depression).

4.2. An uncertainty measure for parametric machines

The sensitivity map provides us with a general indication of how the model is learning from our observations and what the critical points are for each individual. Although very informative for an expert, it cannot be used in a medical context, so it is necessary to develop a strategy to make these interpretations user-friendly. The goal now is to find a way to quantify this level of uncertainty and, most importantly, to determine whether a new observation falls within an uncertain case or a case with a high certainty.

The idea is to initially develop an algorithm that reduces the dimensionality of the sensitivity maps. This will be followed by reducing the cardinality of the data and analyzing it using a graph-based approach. The aim is to cluster the training observations based on their loss function and gain insights into their distribution. To achieve this goal, we utilize UMAP (Uniform Manifold Approximation and Projection) for dimensionality reduction, and Mapper for cardinality reduction.

UMAP algorithm UMAP is a dimensionality reduction algorithm that is used to transform high-dimensional data into a lower-dimensional representation based on manifold learning techniques and ideas from topological data analysis [22]. This algorithm has several advantages over other methods. Firstly, UMAP can preserve both the global and local structure of high-dimensional data sets. This means that it can accurately capture the relationships between points in a lower-dimensional space, while maintaining the important details and structure of the original data. Secondly, it is highly scalable and flexible with various type of data. UMAP constructs an initial high-dimensional graph by creating a fuzzy simplicial complex, which is essentially a weighted graph where edge weights indicate the likelihood of two points being connected. To determine connections, UMAP extends a radius outward from each point and connects points when their radii overlap. However, selecting the appropriate radius is crucial as a too-small radius leads to small, isolated clusters while a too-large radius connects everything together. To address this, UMAP selects a radius locally, based on the distance to each point's nth nearest neighbor, and decreases the likelihood of connection as the radius grows. UMAP also ensures that each point is connected to at least its closest neighbor, preserving the balance between local and global structures. After constructing the high-dimensional graph using the fuzzy simplicial complex, UMAP aims to optimize the layout of a low-dimensional analogue to make it as similar as possible to the high-dimensional graph [22].

Mapper algorithm Mapper is a data analysis and visualization algorithm that is used for topological data analysis (TDA) [14]. First, data are divided into overlapping subsets, known as cover sets. These sets are defined by a filter function that maps each point in the data to a real number. Next, a graph is constructed where the nodes represent the cover sets and the edges represent overlaps between them. The size of the overlap between each pair of sets is measured using a metric such as Jaccard similarity or intersection. The graph is then simplified by collapsing nodes and edges based on a clustering algorithm such

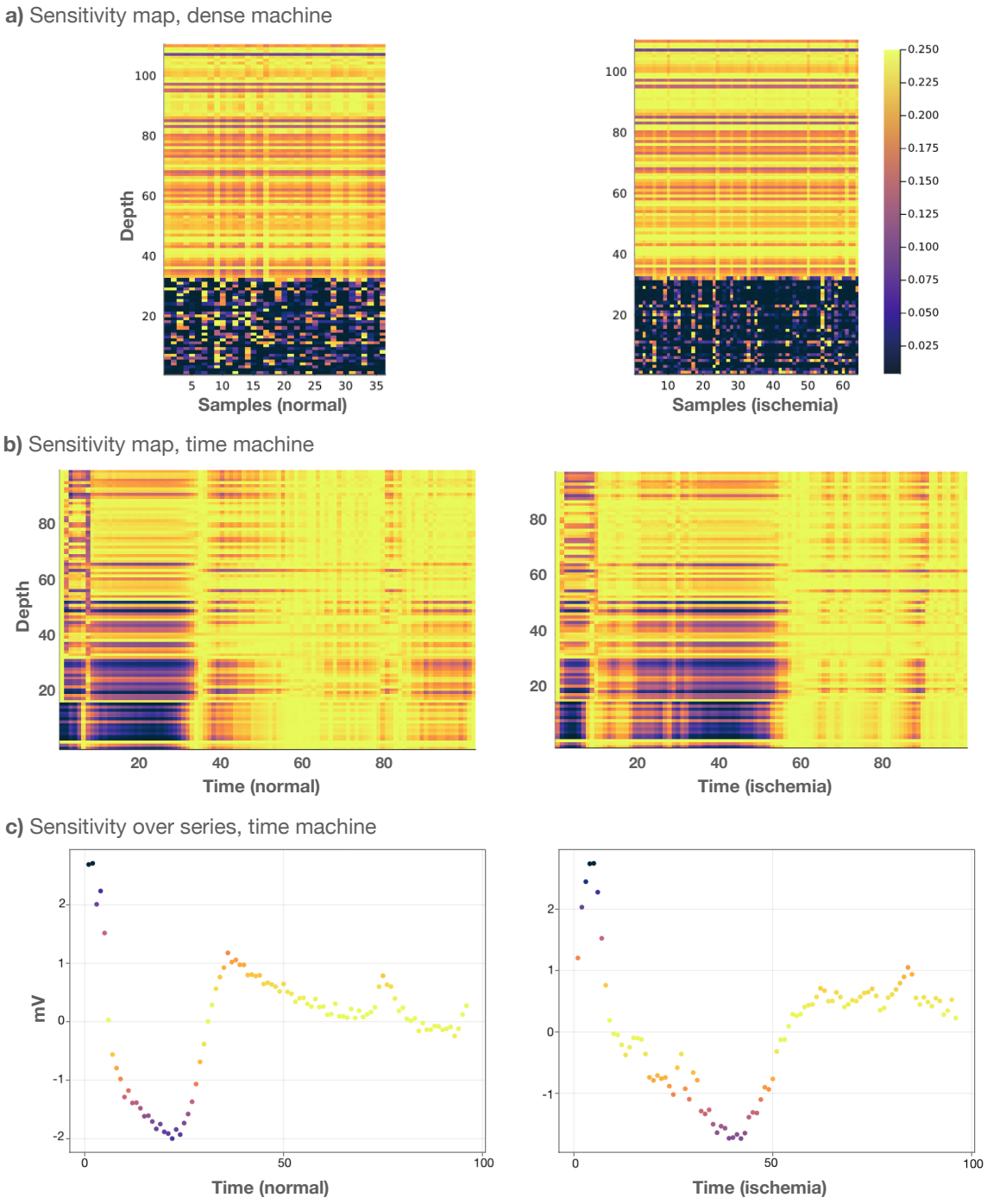

Figure 3: Sensitivity maps represent the values of the derivative of the non-linear activation function on the entire machine architecture. **a)** Sensitivity map generated by training a dense machine. **b)** Sensitivity map for a time machine. **c)** Normal and abnormal heartbeat series colored by sensitivity. We can observe that the most sensitive part of the signal are the peaks in the normal heartbeat that correspond to a ST depression in the ischemia sample and the initial slope.

ID	$\mu(\sigma)$ loss	Ischemia	Accuracy	Precision	Recall	F1 score
1	0.0032 (0.0040)	0.55	0.50	0.50	0.40	0.44
2	0.0009 (0.0014)	0.24	0.76	0.88	1	0.83
3	0.0048 (0.0065)	0.50	0.76	0.86	0.67	0.80
4	0.0045 (0.0039)	0.55	0.77	0.33	0.17	0.40
5	0.0008 (0.0004)	0.90	1	1	0.33	1
6	0.0004 (0.0002)	1	0.94	0	0	0
7	0.0003 (0.0003)	1	0.50	0.50	0.40	0.57
8	0.0006 (0.0007)	0.80	0.61	0.83	0.67	0.74
9	0.0001 (0.0011)	0.61	0.92	0	0	0
10	0.0003 (0.0004)	1	0.78	0.86	0.55	0.77

Table 1: Graph representation parameters regarding fig. 4. Columns represents the ID of the nodes, the mean loss value in each node, the standard deviation in the node, the percentage of ischemia heartbeat's cases in the node, the accuracy on test set, the precision on test set, the recall on test set and the F1 score on test set.

as k-means or hierarchical clustering. This step reduces the complexity of the graph and highlights the most salient features of the data. Finally, the simplified graph is visualized in a lower-dimensional space, such as a 2D plane, using a layout algorithm such as force-directed placement. This allows the user to explore the structure of the data and identify patterns and relationships between the cover sets.

Medical application of sensitivity maps In our case, we reduce the dimensionality of the data using the sensitivity matrices generated from the training data, selecting 40 components for the reduction process. For the Mapper algorithm, we use a simple x-axis projection as the filter function, and used the k-means clustering algorithm with $k = 3$ to identify clusters. We select hyperparameters choosing the best graphical representation, given the fact that all representations shows the same theoretical patterns. The output of the analysis is visualized in fig. 4. The two observed clusters divide the observations based on the degree of loss, providing an indication of their reliability in terms of classification. Specifically, the cluster with lower loss is considered more reliable, while the cluster with greater loss is less reliable. The next step is to identify, given a new observation, which of these two clusters it will belong to, so as to be able to say something about the degree of uncertainty of the classification.

For instance, as shown in fig. 4, there are two distinct cases: a sample that has been correctly classified and appears as green, and a sample that has been misclassified and appears as red. A doctor can evaluate the algorithm's effectiveness by examining cases that fall into an uncertain cluster, such as a cluster with a higher loss, and comparing them to other samples in the same cluster. This recommendation serves as a cautionary note about the reliability of the classification. Once the doctor has been made aware of this, they can examine the image visually and draw more detailed conclusions.

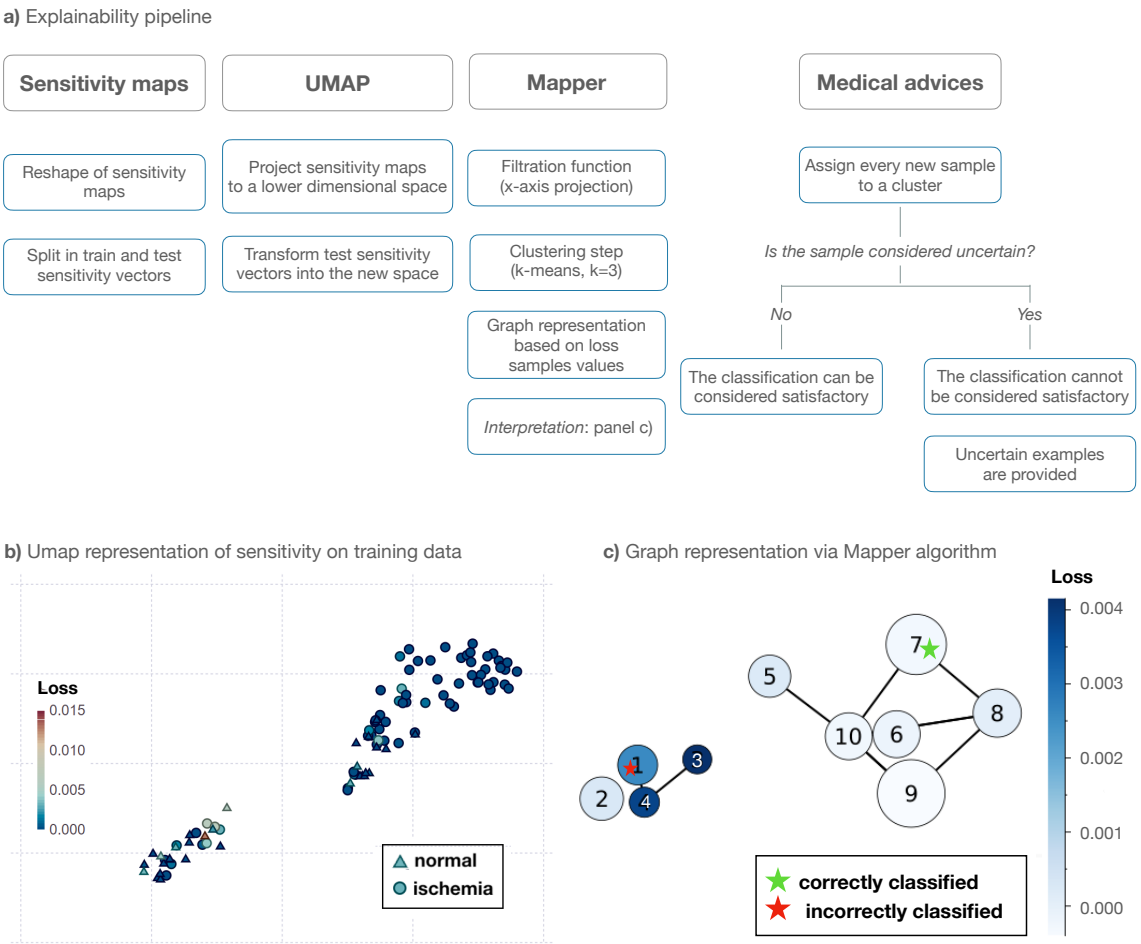

Figure 4: Sensitivity-based confidence measure. **a)** Explainability analysis pipeline, from sensitivity maps to medical advices. **b)** Sensitivity map visualization after dimensionality reduction (UMAP). **c)** Mapper graph on reduced sensitivity maps. The sensitivity maps obtained from the training set are first vectorized and reduced to 40-dimensional points through UMAP. Then, the projected points are clustered via Mapper. The graph presents connected components organized according to the loss realized by the samples associated with their nodes. The green star represents the mapping of a correctly classified test sample according to its sensitivity map (fig. 4, panel c, left). Symmetrically, the red star corresponds to a misclassified sample (fig. 4, panel c, right).

5. Forecast

The second task we try to solve is a forecasting problem regarding energy consumption time series. Every year, global energy consumption continues to rise, making it imperative for energy providers to explore and develop models that can better forecast and plan for energy demand. Accurately predicting the behavior of the energy system is critical in mitigating potential uncertainties and facilitating load shaping, which can help to reduce waste. Moreover, electric energy must be consumed at the same time it is generated in the power plant due to its physical characteristics. Deep neural networks have demonstrated their effectiveness in this regard, as they are capable of learning complex patterns and making precise predictions based on large amounts of historical energy consumption data [25]. This, in turn, can lead to cost savings and improved energy efficiency. Furthermore, accurate energy consumption forecasts can guide decision-making in the energy sector, ensuring a stable and reliable energy supply.

5.1. Dataset

Data structure The dataset is composed by 255 time series about energy consumption, one per user, sampled every 15 minutes. Data were harvested in 2017 in Guayaquil (Ecuador), capturing values for the entire year from January 1st to December 31st. The data is presented in tabular form, consisting of seven columns, with one column indicating the timing, and the other columns representing various types of power usage:

- Timeline;
- Real or Active Power (kW), the power that is actually utilized or consumed;
- Real Power without Power Factor, that is the ratio between Real Power and Total Power;
- Reactive Power (kVA), the power that is developed in the circuit reactance;
- Reactive Power without Power Factor;
- Real Power Demand;
- Reactive Power Demand.

Overall, total energy demand refers to the amount of energy consumed by all appliances, machineries and systems in a particular area or building. Total demand may exceed actual power due to factors such as transmission losses, power factor and other factors affecting energy efficiency. These losses can cause total demand to exceed the actual power delivered to end-users because a minimum value of reactive power is always necessary to maintain constant voltage and supply useful active power.

5.2. Methods

Preprocessing The ultimate objective is to predict real power demand for week 44. To achieve this goal, we conduct a thorough analysis of the data, including data cleaning and standardization. We eliminate 33 users from the analysis due to anomalies such as different time samples or a lack of measurements until the week we want to predict. The differences in distribution among various users are significant, as the areas and energy usage can vary greatly. These differences can have important implications for understanding patterns of energy usage and identifying areas where energy efficiency improvements may be needed. Furthermore, this property will be crucial in enabling us to further explore one of the main

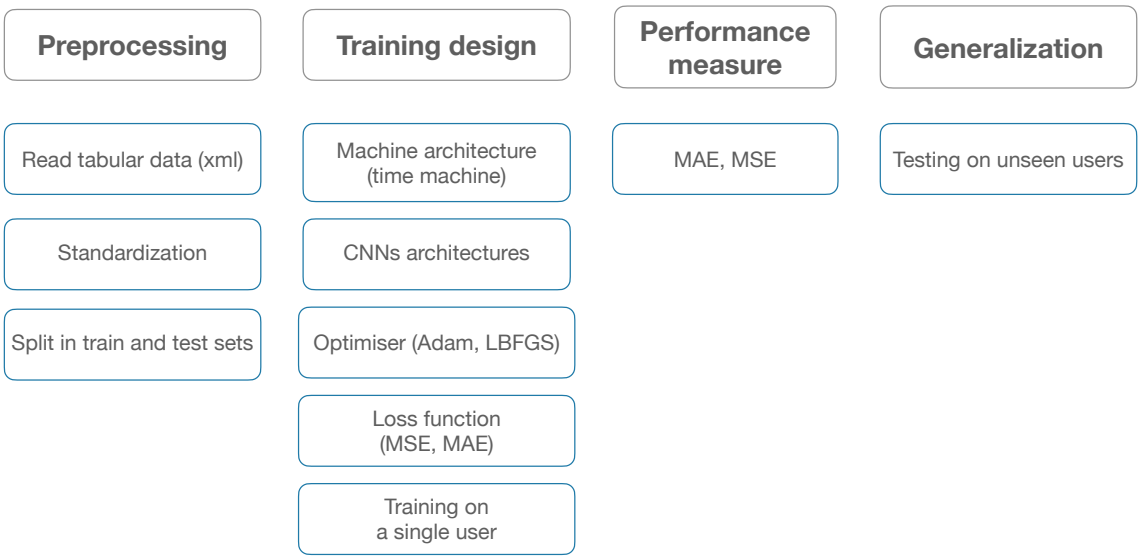

Figure 5: Energy consumption analysis pipeline.

features of parametric machines.

The input data is generated by selecting all demand values except for the last week of the period. The output is created by shifting the input values by one week, which corresponds to the beginning of the second week up to the end. Therefore, the first week of input is not included in the output, and the last week of output is not included in the input. The model is trained using data up to the end of the 43rd week and then evaluated on the 44th week. The reason for choosing the 44th week for evaluation is that not all users have demand values in the last weeks of the year. Finally, we split the dataset into training and testing sets.

Architectures Unlike the ischemia dataset, we don't have a state of the art to compare with parametric machine, so we implement three different models: a time machine and two classical convolutional neural networks (CNNs).

After experimenting with various combinations of dimensions, we determine that dividing the space into four subspaces, each with a depth of four and one with a depth of one, yielded the best results. This approach results in a total of 13 subspaces. We utilize a time block of two days. Additionally, to ensure the preservation of a full day, we incorporate a padding value of 96, which corresponds to 24 hours with 4 observations every hour. For nonlinearity, we implement the sigmoid function. The model also includes a convolutional layer with a kernel size of one, which accepts 13 channels as inputs representing the subspaces, and generates a single output channel.

The second model we have defined is a CNN consisting of 4 convolutional layers (CNN-1). The architecture is composed of a sequence of convolutional layers that are connected in a sequential manner without any shortcuts. We carefully selected the architecture of the convolutional model to resemble the structure of the time machine so that a fair comparison could be made between the two models. We did not use a recurrent model, which is typically used for time series, because a different data arrangement is required, which would not be compatible with the time machine. The convolutional neural network utilized for this task is a one-dimensional CNN, as the input data consists of time series. We chose four layers because this corresponds to the number of subspaces in which the time machine is divided.

We chose a one-day filter as kernel, which consists of 24 hours with four observations every 15 minutes in an hour. Regarding the padding setting, we added 95 zero-values to the left side (representing the past) of the input data, and zero-values to the right side (representing the future).

Both models share the same parameter settings with the aforementioned configurations. However, there is a fundamental difference between them. In the sequential model, the last layer only takes the output of the previous layer as input without any shortcuts, while in the time machine, the last layer takes all the previous layer's output using all possible shortcuts as input. Nevertheless, the number of parameters in the sequential model and the time machine cannot be directly compared. To make the number of parameters comparable, we attempted to increase the number of parameters in the CNN. In the second architecture version of the CNN (CNN-2), which still contains four convolutional layers, we arrange the layers hyperparameters in order to have a comparable number of model parameters with the time machine. We retain all other settings from the first convolutional model. This change results in a total of 16923 parameters, allowing us to make a fair comparison. All of these architectures are implemented in [13].

Training and testing In order to test the generalization power of both classical and time machine models, we decided to train our models on a single user due to the high electrical demand variance across users discussed in fig. 6. Even though this strategy may not capture the full complexity of the dataset, it allows us to compare the performance of all models not only on the test set associated with the selected user but also on all users. By doing so, we can obtain a more encompassing view of how parametric machines and convolutional neural networks generalize on unseen, highly variable data.

During the training process, we use the mean squared error (MSE) and the mean absolute error (MAE) as loss functions. Regarding the optimization step, we choose to work with the LBFGS optimizer.

5.3. Results

The parametric machine model showed better performance than CNN-1 model even with a lower number of epochs. CNN-2, which had additional parameters compared to the first model, performed slightly better in both training and testing for the trained user.

After training, we evaluate the models prediction power showing how models predict on the trained user, as we can see in fig. 6.

The results show that the parametric machine outperforms the CNNs in terms of predictions accuracy in all users. The CNN models seem to predict a sinusoidal trend, while the parametric machine is able to accurately predict the real trend during time. This suggests that the parametric machine is better able to capture the underlying patterns in the data and generalize to new, unseen data (we will discuss more on this in section 6).

6. Generalization

Our goal has always been to uncover general patterns that enable us to make accurate predictions on new examples drawn from the same underlying population [17]. The ability of a deep learning model to generalize to new, unseen data is evaluated through the use of validation and test sets. During the training process, the model is optimized on the training set, while the validation set is used to monitor its performance on data that it has

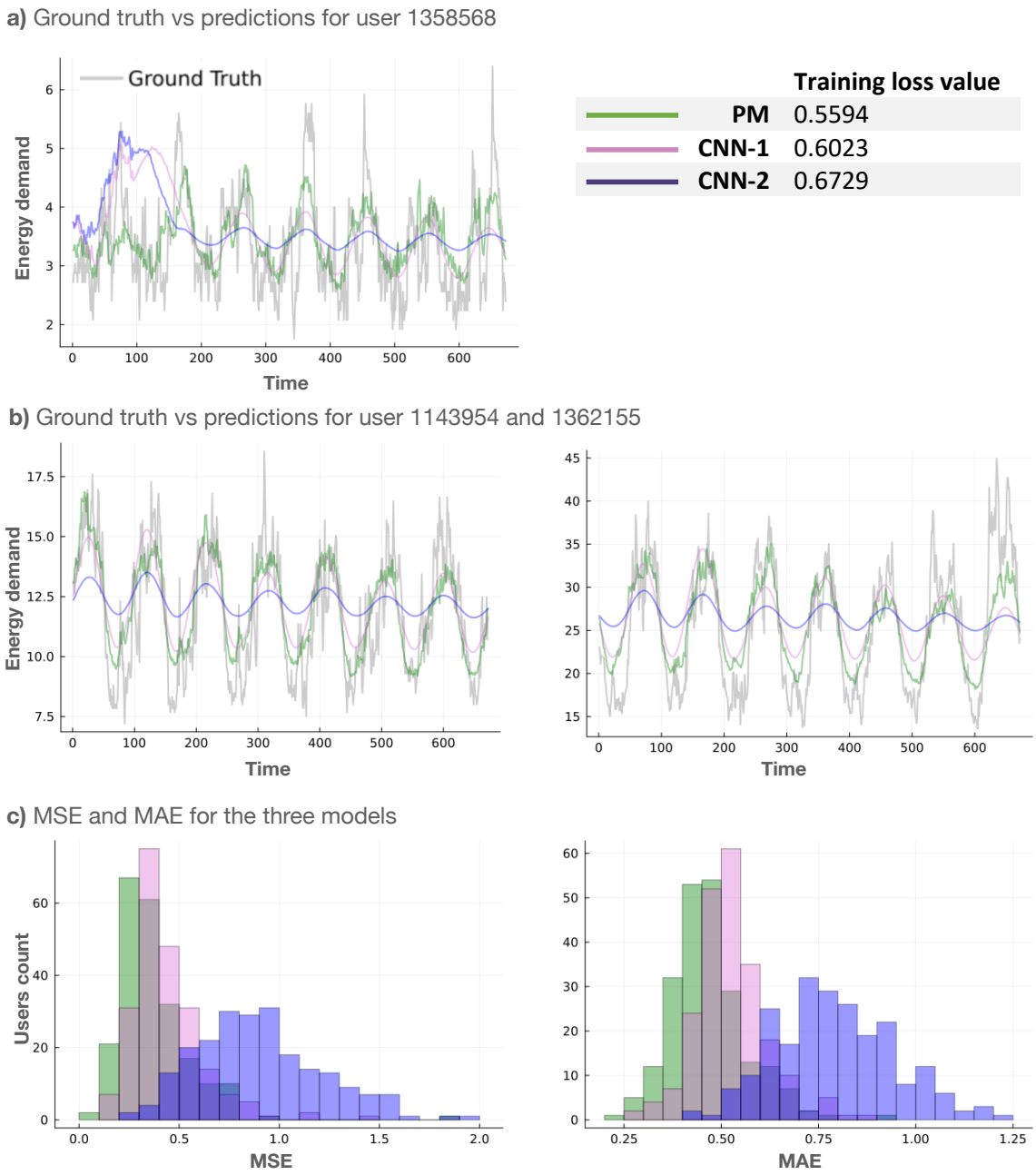

Figure 6: Data representation, train loss and prediction for a single user. **a)** Ground thruth for the 44th week and predictions for the three models. Parametric machine is able to predict accurately the behaviour of the energy demand, the two CNNs tend to predict a sinusoidal trend over time. **b)** Two new user's predictions example using models trained on a single different user. Parametric machines perform well also on new users showing their generalization power. **c)** MSE and MAE barplot showing that the parametric machine model performs better then the other models in the majority of the observations.

not been trained on. By adjusting the model's hyperparameters based on the validation set's performance, we can help prevent overfitting and improve the model's ability to generalize. Once the model has been optimized, it is evaluated on a separate test set that has not been used in either the training or validation process. This final evaluation provides a measure of the model's ability to generalize to new data. Additionally, statistical measures such as bias and variance can also provide insights into a model's generalization capabilities. A model with high bias may underfit the training data and perform poorly on both the training and validation sets, while a model with high variance may overfit the training data and perform well on the training set but poorly on the validation and test sets. By balancing bias and variance, we can improve a model's ability to generalize to new data. In addition, there are also techniques that can be used to improve a deep learning model's generalization properties. One such technique is regularization. Other techniques include dropout, early stopping, data augmentation, and transfer learning.

It is also worth noting that the generalization properties of a deep learning model can be influenced by the quality and quantity of the training data [20]. Having a diverse and representative training set can help the model learn more generalized patterns and improve its ability to generalize to new data. Ultimately, the ability of a deep learning model to generalize is a key factor in its effectiveness and applicability to real-world problems [3]. As such, understanding and improving the generalization properties of deep learning models is an important area of research in machine learning.

Generalization in energy context In section 5, the study led us to observe a fundamental characteristic of parametric machines: their capacity for generalization. Initially, the study focused on individual users due to computational issues, but it ultimately led to a surprising discovery. By analyzing individual users, our model demonstrated generalization abilities that extended to all other users, resulting in highly accurate predictions.

In this energy consumption forecasting problem, the results indicate that the time machine outperforms convolutional architectures. The machine's capacity to process all inputs and retain more information through the presence of shortcuts allows for better detection of fluctuations. This architecture is more effective in capturing the complex and varied patterns in the data, giving the machine an advantage over convolutional neural networks in energy consumption forecasting.

In fact, the results demonstrate that the time machine is capable of generalizing data with different magnitudes (e.g., user 1362155) and greater oscillations, as evidenced by the prediction examples of different users, as seen in fig. 6.

To assess the overall performance of the models, we generated a distribution plot of the mean absolute error (MAE) and mean squared error (MSE) for each user in fig. 6. In both cases, the parametric machine consistently demonstrated the lowest mean and standard deviation in the error distribution. This finding suggests that the other models are less accurate than the machine in energy consumption case.

7. Conclusion

We consider a novel deep-learning framework—parametric machines—that generalizes deep neural architectures and provides a formal mathematical definition of operators and models commonly used in deep learning. There, classical and novel neural architectures are described as points of a function space. Among these points, we find novel architectures with a rich shortcut structure that, in line with recently described hand-crafted models,

e.g., [8, 15, 21], could enable us to overcome issues such as vanishing and exploding gradient or instability.

To test the hypothesis mentioned above, we apply parametric machines to two case studies. The two proposed case studies are, by their very own nature, challenging for classical deep-learning models. First, we test parametric machines on a time series classification task with a very limited training set; then, we investigate a time series forecasting application where the model needs to infer periodicity at several time granularities and predict 672 time steps.

In the first application, namely classification of the ECG samples from ECG200, we show that parametric machines outperform the current state of the art. We devise and implement an explainability module—sensitivity maps—that takes advantage of the formal definition of parametric machines. Since machines are endofunctions on a global space, we compute the derivative of the nonlinear activation function on the linearized machine. On the one hand, sensitivity maps allow us to highlight parts of the signal that are relevant for the model. Hence, sensitivity maps could be useful to the end-user to gain intuition on the model's decisions. On the other hand, via a dimensionality and cardinality reduction algorithm, we provide a measure of classification uncertainty on test data. We believe that this type of approach can be complementary to approaches such as saliency maps [1], with a strong difference in that sensitivity maps consider the entire architecture rather than only the input layer.

In the second application, we compared a parametric machine with its classical sequential counterparts on an energy-consumption forecast. Parametric machines better grasp multiscale periodicities of the signal yielding more accurate long-term predictions. This application enables us to explore the generalization capability of parametric machines. By training parametric machines and classical models on a single user and testing them on the other 221 users, we show that the former model better generalizes to unseen, out-of-distribution data.

While this article has directly addressed the two open problems mentioned above, namely explainability and generalization, further research can explore and discover additional properties such as adaptability [10]. Adaptability refers to the ability of a model to adjust its parameters in response to changes in the input data. This means that the model is able to learn and improve its performance over time, as it is exposed to more data, making it useful in a wide range of applications, such as recommendation or fraud detection.

In a forthcoming paper, we plan to further explore the potential of these models in different fields and applications, which could lead to the development of more accurate, reliable, and explainable deep learning solutions.

References

[1] Nishanth Thumbavanam Arun, Nathan Gaw, Praveer Singh, Ken Chang, Katharina Viktoria Hoebel, Jay Patel, Mishka Gidwani, and Jayashree Kalpathy-Cramer. Assessing the validity of saliency maps for abnormality localization in medical imaging. *arXiv preprint arXiv:2006.00063*, 2020.

[2] Viktor Axillus. Comparing julia and python: An investigation of the performance on image processing with deep neural networks and classification, 2020.

[3] Pietro Barbiero, Giovanni Squillero, and Alberto Tonda. Modeling generalization in machine learning: A methodological and computational study. *arXiv preprint arXiv:2006.15680*, 2020.

[4] Paul Barham and Michael Isard. Machine learning systems are stuck in a rut. In *Proceedings of the Workshop on Hot Topics in Operating Systems*, pages 177–183, 2019.

[5] Mattia G. Bergomi and Pietro Vertechi. `https://github.com/LimenResearch/ParametricMachinesDemos.jl`, 2022.

[6] Jeff Bezanson, Stefan Karpinski, Viral B Shah, and Alan Edelman. Julia: A fast dynamic language for technical computing. *arXiv preprint arXiv:1209.5145*, 2012.

[7] Alexander Binder, Sebastian Bach, Gregoire Montavon, Klaus-Robert Müller, and Wojciech Samek. Layer-wise relevance propagation for deep neural network architectures. In *Information science and applications (ICISA) 2016*, pages 913–922. Springer, 2016.

[8] Nico Curti. Implementation and optimization of algorithms in biomedical big data analytics. 2020.

[9] Arun Das and Paul Rad. Opportunities and challenges in explainable artificial intelligence (xai): A survey. *arXiv preprint arXiv:2006.11371*, 2020.

[10] Nikolaos Doulamis. Adaptable deep learning structures for object labeling/tracking under dynamic visual environments. *Multimedia Tools and Applications*, 77:9651–9689, 2018.

[11] Andre Esteva, Brett Kuprel, Roberto A Novoa, Justin Ko, Susan M Swetter, Helen M Blau, and Sebastian Thrun. Dermatologist-level classification of skin cancer with deep neural networks. *nature*, 542(7639):115–118, 2017.

[12] Martina Garavaglia. `https://github.com/martina-garavaglia-sdg/ischemia_classification_analysis.jl`, 2023.

[13] Martina Garavaglia and Paola Serra. `https://github.com/paola-serra-sdg/Energy_forecast.jl`, 2023.

[14] Boris Goldfarb. The mapper algorithm and its applications. In *15th Annual Workshop on Topology and Dynamical Systems*, 2018.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[16] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, pages 646–661. Springer, 2016.

[17] Kenji Kawaguchi, Leslie Pack Kaelbling, and Yoshua Bengio. Generalization in deep learning. *arXiv preprint arXiv:1710.05468*, 2017.

[18] Richard E. Klabunde. `https://www.cvphysiology.com/`.

[19] Antonio Lavecchia. Deep learning in drug discovery: opportunities, challenges and future prospects. *Drug discovery today*, 24(10):2017–2032, 2019.

[20] Jaeeun Lee, Hanme Jang, Jonghyeon Yang, and Kiyun Yu. Machine learning classification of buildings for map generalization. *ISPRS International Journal of Geo-Information*, 6(10):309, 2017.

[21] Tianyi Liu, Minshuo Chen, Mo Zhou, Simon S Du, Enlu Zhou, and Tuo Zhao. Towards understanding the importance of shortcut connections in residual networks. *Advances in neural information processing systems*, 32, 2019.

[22] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

[23] Riccardo Miotto, Fei Wang, Shuang Wang, Xiaoqian Jiang, and Joel T Dudley. Deep learning for healthcare: review, opportunities and challenges. *Briefings in bioinformatics*, 19(6):1236–1246, 2018.

[24] Shubham Mittal and Yasha Hasija. Applications of deep learning in healthcare and biomedicine. *Deep learning techniques for biomedical and health informatics*, pages 57–77, 2020.

[25] Elena Mocanu, Phuong H Nguyen, Madeleine Gibescu, and Wil L Kling. Deep learning for estimating building energy consumption. *Sustainable Energy, Grids and Networks*, 6:91–99, 2016.

[26] Georgios Z Papadakis, Apostolos H Karantanas, Manolis Tsiknakis, Aristidis Tsatsakis, Demetrios A Spandidos, and Kostas Marias. Deep learning opens new horizons in personalized medicine. *Biomedical reports*, 10(4):215–217, 2019.

[27] Francisco Romero-Ferrero, Mattia G Bergomi, Robert C Hinz, Francisco JH Heras, and Gonzalo G De Polavieja. Idtracker. ai: tracking all individuals in small or large collectives of unmarked animals. *Nature methods*, 16(2):179–182, 2019.

[28] Moussa Saleh and John A Ambrose. Understanding myocardial infarction. *F1000Research*, 7, 2018.

[29] Sam Sattarzadeh, Mahesh Sudhakar, Konstantinos N Plataniotis, Jongseong Jang, Yeonjeong Jeong, and Hyunwoo Kim. Integrated grad-cam: Sensitivity-aware visual explanation of deep convolutional networks via integrated gradient-based scoring. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1775–1779. IEEE, 2021.

[30] Dinggang Shen, Guorong Wu, and Heung-Il Suk. Deep learning in medical image analysis. *Annual review of biomedical engineering*, 19:221–248, 2017.

[31] Amitojdeep Singh, Sourya Sengupta, and Vasudevan Lakshminarayanan. Explainable deep learning models in medical image analysis. *Journal of Imaging*, 6(6):52, 2020.

[32] Pietro Vertechi and Mattia G Bergomi. Machines of finite depth: towards a formalization of neural networks. *arXiv preprint arXiv:2204.12786*, 2022.

[33] Patrick Weber, K Valerie Carl, and Oliver Hinz. Applications of explainable artificial intelligence in finance—a systematic review of finance, information systems, and computer science literature. *Management Review Quarterly*, pages 1–41, 2023.

[34] Xujing Yao, Xinyue Wang, Shui-Hua Wang, and Yu-Dong Zhang. A comprehensive survey on convolutional neural network in medical image analysis. *Multimedia Tools and Applications*, pages 1–45, 2020.

A. Julia programming language

The programming language chosen to pursue this project is Julia. While it is not yet widely recognized in the data science field, Julia has significant advantages in terms of performance and usability [6]. Julia is a flexible and dynamic language that is especially well-suited for scientific and numerical computing. Furthermore, it boasts a performance level that is comparable to that of more traditional statically-typed languages.

The decision to use Julia as the programming language for this project was not arbitrary, but based on its high performance and potential for exploration, compared to other languages. A brief comparison between Julia and the better-known Python language can help illustrate this choice [2]. In terms of speed, Julia is much faster than Python, with an execution speed that is close to that of C. From a community perspective, Python has been around for a long time and has a large network of programmers, making it easier to find online solutions to problems. However, Julia code can be easily converted to Python, while the reverse is not true. While Python has a large number of advanced libraries, Julia can interface with C and Fortran libraries to handle tasks that have not yet been implemented. Lastly, Julia is dynamically typed and allows for code development without specifying the type of object being used, but as previously mentioned, type declarations are what make Julia highly efficient. Python is also dynamically typed, but lacks the benefits of type declaration.

The two main packages that we have used in this project are Flux and Optim. In Julia, Flux is a machine learning package for building and training neural networks. It provides a set of high-level abstractions and utilities for defining, training, and evaluating neural network models. Flux allows users to define neural network models using a simple and intuitive syntax, and provides a wide range of layer types and activation functions that can be easily combined to create complex models. It also supports various types of loss functions, optimizers, and regularization techniques for training these models. One of the key features of Flux is its support for automatic differentiation, which allows users to define custom loss functions and backpropagation algorithms for their models. This makes it easy to build and train complex neural network models with minimal boilerplate code. Flux also provides a range of utilities for working with data, including loading and preprocessing data, splitting data into training, validation, and test sets, and data augmentation techniques such as random cropping and flipping.

Optim is a package for optimization algorithms. It provides a wide range of optimization algorithms, including unconstrained optimization, constrained optimization, global optimization, and derivative-free optimization. Optim allows users to define objective functions and constraints using a simple and flexible syntax, and provides a range of optimization algorithms that can be easily applied to these functions. It also supports various types of constraints, including linear and nonlinear equality and inequality constraints, and provides utilities for working with sparse matrices and nonlinear functions. One of the key features of Optim is its support for automatic differentiation, which allows users to define custom objective functions and gradients for their optimization problems. This makes it easy to optimize complex functions with minimal boilerplate code.

Abstract in lingua italiana

La mancanza di una definizione formale rende l'implementazione di reti neurali un compito altamente specializzato e che richiede molto tempo. In questo articolo, abbiamo proposto due casi studio riguardanti la classificazione e la predizione di serie temporali utilizzando le macchine parametriche, una generalizzazione formale di reti neurali. Nel corso di questi casi studio, abbiamo dimostrato come le macchine parametriche siano in grado di competere e di conseguire prestazioni migliori rispetto ai corrispondenti omologhi classici, in particolare in un compito di classificazione di elettrocardiogrammi. In aggiunta, abbiamo introdotto una tecnica di regolarizzazione per le macchine parametriche e un algoritmo finalizzato al calcolo di una nozione di incertezza. Abbiamo poi applicato le macchine parametriche ad un problema di forecasting riguardante consumo di energia elettrica, confrontando le performance con quelle ottenute da architetture moderne, dopo averli resi comparabili a livello di struttura, per poi investigare sulla capacità di generalizzazione di questi modelli.

Parole chiave: Macchine parametriche, serie temporali, intelligibilità, sensibilità, generalizzazione

Acknowledgements

First and foremost, I would like to express my gratitude to professor Secchi for giving me the opportunity to work on a project in a new and unfamiliar research field for him. I am deeply appreciative of his interest in everything I proposed and for his guidance throughout this journey.

I would also like to extend my thanks to Mattia and Pietro, my co-advisors, who have been instrumental in this thesis with their unwavering patience and eagerness to teach. Their constant support and assistance has been invaluable. I would also like to extend my gratitude to Maurizio for being the first to take an interest in my thesis needs and for consistently imparting his knowledge and wisdom to me.

I am deeply grateful to my family, especially my father Giovanni, mother Cinzia and sister Valeria, who have been my rock through the years, offering their support and encouragement during both the high and low moments of this journey.

I would also like to express my gratitude to my little nephew Alessandro and my brother in law Andrea, who brought joy and laughter to my days.

A special thank you to my lifelong best friend Valeria, who has always been there for me, wherever she is and whatever she does.

I would like to extend my gratitude to my friend Alice, for always being present and providing comfort with her sweet words.

I would like to express my thanks to Virginia, Francesca, Ilaria, Federica and Alessandro, my university mates who shared many memorable moments of study and laughter throughout this degree program.

Thank you to all my friends at JustZebrallo, the two Andrea, Claudia, Erika, Giacomo, Mattia and Laura, for always being interested in me and for providing support.

I would also like to express my gratitude to my new friends Daniele and Paolo, with whom I enjoy playing beach volley and spending evenings filled with laughter.

I am also thankful to all my colleagues at SDG for making me feel welcome and at ease

from day one. A special thank you to Paola, who has been a constant source of support during this project.

I would also like to express my gratitude to Simone who, despite our recent acquaintance, listened to me when I needed it the most. Thanks for showing me that coming in last place is just as valuable as coming in first.

Lastly, I would like to thank everyone who has supported me on this journey, but especially to those who challenged me and helped shape me into the person I am today.