Department of Economics- FEA/USP

Deep Haar Scattering Network in Unidimensional Pattern Recognition Problems

FERNANDO FERNANDES NETO
CLAUDIO GARCIA
RODRIGO DE LOSSO DA SILVEIRA BUENO
PEDRO DELANO CAVALCANTI
ALEMAYEHU SOLOMON ADMASU

WORKING PAPER SERIES Nº 2019-16

DEPARTMENT OF ECONOMICS, FEA-USP
WORKING PAPER Nº 2019-16

Deep Haar Scattering Networks in Unidimensional Pattern Recognition Problems

Fernando Fernandes Neto (fernando_fernandes_neto@usp.br)

Claudio Garcia

Rodrigo de Losso da Silveira Bueno (delosso@usp.br)

Pedro Delano Cavalcanti

Alemayehu Solomon Admasu

Abstract:

The aim of this paper is to discuss the use of Haar scattering networks, which is a very simple architecture that naturally supports a large number of stacked layers, yet with very few parameters, in a relatively broad set of pattern recognition problems, including regression and classification tasks. This architecture, basically, consists of stacking convolutional filters, that can be thought as a generalization of Haar wavelets, followed by nonlinear operators which aim to extract symmetries and invariances that are later fed in a classification/regression algorithm. We show that good results can be obtained with the proposed method for both kind of tasks. We outperformed the best available algorithms in 4 out of 18 important data classification problems, and obtained a more robust performance than ARIMA and ETS time series methods in regression problems for data with invariances and symmetries, with desirable features, such as possibility to evaluate parameter stability and easy structural assessment.

Keywords: Haar Scattering Network; Pattern Recognition; Classification; Regression; Time Series.

JEL Codes: C38; C45; C52; C63

DEEP HAAR SCATTERING NETWORKS IN UNIDIMENSIONAL PATTERN RECOGNITION PROBLEMS

A PREPRINT

Fernando Fernandes Neto
Department of Business, Accounting and Economics
University of São Paulo
Cidade Universitária, 05508-010, São Paulo, Brazil
fernando_fernandes_neto@usp.br

Alemayehu Solomon Admasu
Department of Physics & Astronomy
Rutgers University
Piscataway, New Jersey, 08854, USA
a.solomon@rutgers.edu

Rodrigo de Losso
Department of Business, Accounting and Economics
University of São Paulo
Cidade Universitária, 05508-010, São Paulo, Brazil
delosso@usp.br

Claudio Garcia
Department of Electrical Engineering
University of São Paulo
Cidade Universitária, 05508-010, São Paulo, Brazil
clgarcia@lac.usp.br

Pedro Delano Cavalcanti
Department of Physics & Astronomy
Rio de Janeiro State University
R. São Francisco Xavier - 524, 20559-900, Rio de Janeiro, Brazil
pedelano@gmail.com

May 15, 2019

ABSTRACT

The aim of this paper is to discuss the use of Haar scattering networks, which is a very simple architecture that naturally supports a large number of stacked layers, yet with very few parameters, in a relatively broad set of pattern recognition problems, including regression and classification tasks. This architecture, basically, consists of stacking convolutional filters, that can be thought as a generalization of Haar wavelets, followed by non-linear operators which aim to extract symmetries and invariances that are later fed in a classification/regression algorithm. We show that good results can be obtained with the proposed method for both kind of tasks. We outperformed the best available algorithms in 4 out of 18 important data classification problems, and obtained a more robust performance than ARIMA and ETS time series methods in regression problems for data with invariances and symmetries, with desirable features, such as possibility to evaluate parameter stability and easy structural assessment.

***K*eywords** Haar Scattering Networks · Pattern Recognition · Classification · Regression · Time Series

1 Introduction

Pattern recognition in time-series is a fundamental data analysis type for understanding dynamics in real-world systems. It is common to gather time-series data from a wide range of problems, such as stock market prediction, speech and music recognition, motion capture data and electronic noise data (19). They can also be obtained by means of successive measurements of higher dimensional problems, such as image contours, sequential counts from network nodes and other mathematical objects, as can be seen in (1).

Analysis of time-series data has been the subject of active research for decades and many approaches for modeling them have been developed. Traditional methods, for instance autoregressive models, Linear Dynamical Systems and Hidden Markov Models (HMM) need an experienced modeler to identify and estimate them, besides the fact that they are subject to failures in modeling accurately complex real-world data (19).

To circumvent these limitations, machine learning based methods became an attractive solution to data analysis of this kind, because they can be applied to linear and non-linear systems and are able to extract features (which can also describe system states) in both Euclidean and non-Euclidean domains, allowing a significant performance gain, as can be seen in (20).

In this context, in order to increase feature extraction capabilities, machine learning methods have become deeper and deeper, where the most prominent deep learning methods are Convolutional Neural Networks (CNNs). They are employed in a wide range of tasks such as text classification, natural language processing, image processing and time-series data modeling (11).

CNNs basically consist of multiple convolutional filters, that act as trainable layers, stacked on top of each other and usually followed by a non-linear operator and a pooling layer, followed by a classification algorithm on its tip (20). It is important to mention that CNNs also overcome a prevalent problem in most Artificial Neural Networks (ANN), which is the lack of understanding of the underlying statistical and geometric features extracted from the analyzed signal, making the comprehension of why an ANN makes a particular decision a difficult task (4; 6).

In the quest of trying to understand the success of these algorithms, (7), (24) and (6) have identified that symmetries and invariances play a fundamental role in feature extraction, given that relevant information contained in a wide range of different signals (such as sounds or images) are typically not affected by translations or rotations and are stable to deformations.

Also, (7) suggests that less flexible feature extractors can be obtained by means of simple convolutional filters such as wavelets, followed by simple non-linear operators, yet yielding very good results, despite its simplicity. The key factor of this architecture is the preservation of some important properties of the traditional deep networks, while allowing the reduction of the computational complexity.

Complementing this work and dealing with only Haar Scattering Transforms, which are the simplest Scattering Convolutional Transforms, (9) shows that it is possible to solve traditional classification problems, such as digit recognition, with surprisingly greater mathematical/computational simplicity.

In that way, (11) extended this work for 1D signal analysis such as time-series data, showing that general-purpose approximator functions can be obtained based on Haar Scattering Networks, where, for demonstration purposes, only simple Ordinary Least Squares (OLS) regressors were used, with the absolute value function as the non-linear operator.

Having contextualized our research, the main idea of the present paper is to extend (11) by feeding extracted features into classifiers and regressors (such as Support Vector Machines (SVM), OLS regressors and Random Forests) in order to classify/forecast different kinds of signals, using different non-linear operators and an optional pooling layer, which extracts statistical properties of the features, allowing a richer mapping, as can be seen in (3).

We intend to demonstrate that using a very simple architecture, with a relatively large number of stacked layers and a very few parameters, can exhibit very good results - even improving some known results about important problems. These results may open ways to the development of new Automatic Machine Learning (AutoML) algorithms, which is a very recent research field, that aims to find the best performing learning algorithm with minimal human intervention, that is, to automate the design choices of the network (such as topology, optimization procedure, regularization, stability methods) by using hyperparameter optimization (30), with computational simplicity.

Moreover, we also intend to show that this architecture allows a better understanding of key discussions related to time series problems (but not restricted to them) in terms of structural stability of the model and how hyperparameters may change according to changes in the samples, to verify how stable is the identified model. Having very few parameters enables a simple and direct comparison of the models, and increases the number of degrees of freedom, which turns this architecture less subject to over-fitting when dealing with small samples.

Finally, in the Appendix, we also show that despite having a very small number of hyperparameters, our approach still allows to generate a very rich and wide set of different convolutional / filtering operations, providing a very interesting way to feed the well known families of machine learning algorithms.

2 Theory

2.1 Wavelet Transforms

Fourier transforms have many applications in science and engineering, and in the realm of time-invariant signals they provide simple and effective answers to most questions. On the other hand, they become very ineffective with non-stationary problems, due to the fact that sine and cosine functions are just localized in terms of their frequency. They are non-localized in time. In order to solve this problem, a viable substitute for Fourier are Wavelet Transforms.

A discrete wavelet transform is a transform whose basis is composed of a family of orthonormal functions ψ, called wavelets, allowing to capture both frequency and location (time and space), unlike classical Fourier Transform (22). A Haar wavelet is a particular type of wavelet that is used as the orthonormal basis of the Haar Scattering Network. It is defined by a function ψ, as follows.

$$\psi(t) = \begin{cases} 1, & \text{if } 0 < t \leq 1/2 \\ -1, & \text{if } 1/2 < t \leq 1 \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Its respective scaling function, Ψ, is given by:

$$\Psi(t) = \begin{cases} 1, & \text{if } 0 < t \leq 1 \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

It is possible to derive, from the definition of a Haar wavelet and wavelet transforms, a pair of equations for calculating the coefficients of the Haar Wavelet Transform, as reviewed in (11):

$$\begin{aligned} \chi_\omega(k,n) = 2^{-1/2}(&\chi_\omega(2k, n+1) + \\ &\chi_\omega(2k+1, n+1)) \end{aligned} \tag{3}$$

$$\begin{aligned} X_\omega(k,n) = 2^{-1/2}(&\chi_\omega(2k, n+1) - \\ &\chi_\omega(2k+1, n+1)). \end{aligned} \tag{4}$$

For more information about the mathematical definitions and properties of Haar wavelet transforms, see (22) and (7).

2.2 Haar Scattering Networks

Scattering networks were introduced as convolution networks, computed with iterated wavelet transforms, to obtain invariants which are stable to deformations (23; 9).

A Haar Scattering Network was originally defined in (8) and (9) by a sequence of layers, which operates over an input positive d-dimensional signal $x \in (\mathbb{R}^d)^+$. The general scheme of Haar Scattering Networks is to iteratively extract Wavelet coefficients of the signal and apply a point-wise absolute value operator on them.

As seen on (8) a Haar scattering is calculated by iteratively applying the following permutation invariant operator:

$$(\alpha, \beta) \to (\alpha + \beta, |\alpha - \beta|). \tag{5}$$

The values α and β can be recovered by Equations (6) and (7) enabling to reconstruct the whole previous layer values if α and β are real positive:

$$max(\alpha, \beta) \to \frac{1}{2}(\alpha + \beta + |\alpha - \beta|) \tag{6}$$

$$min(\alpha, \beta) \to \frac{1}{2}(\alpha + \beta - |\alpha - \beta|). \tag{7}$$

The network layers are defined as 2D arrays $S_j x(n, q)$ with dimensions $2^{-j} d \cdot 2^j$, where n is a node number and q denotes a feature index. It follows that S_j is a permutation invariant operator that acts over a set of nodes calculated in the previous layer with Equations (8) and (9).

$$S_{j+1}x(n, 2q) \rightarrow S_j x(a_n, 2q) + S_j x(b_n, 2q) \tag{8}$$

$$\begin{aligned} &S_{j+1}x(n, 2q+1) \rightarrow \\ &\qquad |S_j x(a_n, 2q) - S_j x(b_n, 2q)| \end{aligned} \tag{9}$$

where a_n and b_n work as optimizable maps of pairs, dependent on the features extracted.

The iterative extraction of wavelets coefficients of the signal and the application of absolute point-wise operators can be seen in Figure 1.

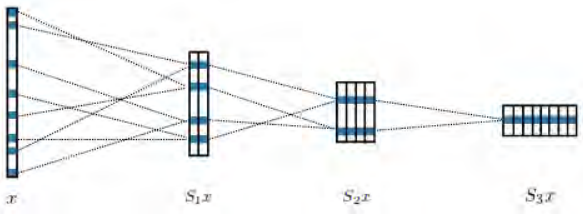


Figure 1: A Haar scattering network computes each coefficient of a layer $S_{j+1}x$ by adding or subtracting a pair of coefficients in the previous layer ($S_j x$) (8).

Pairing rules (a_n, b_n) are optimized as in Algorithms 3 and 4 by either the method of Nelder & Mead (26) or Grid search, so that we obtain scale and shift parameters, σ and τ, respectively. In the current implementation, they act in a signal of length N, where $a_n = n$ and $b_n = (2^{1-j} \cdot N \cdot \sigma) + \tau + n$.

Therefore, these pairing rules differ from the traditional Haar filtering scheme by treating 1D signals as entities that can be represented as graphs, where each node represents a system state, which is directly connected to other states due to their respective multiscale geometric features and invariances, which themselves arise due to possible factors, such as periodicities and trends, that are usually reflected in their spectral or frequency properties.

The main similarities with (11) cease here. The key idea of this work is to extend this approach with other non-linear operators (in addition to pointwise absolute value operators) and explore real regression and classification problems using SVMs, OLS regressors as well as random forests.

To achieve this goal, it is also worth mentioning that Haar Scattering Networks, as presented in (8), have interesting properties that should be kept when modifying the non-linear operators: the capacity to capture both frequency and location; the convenient information compression initially provided by the absolute value operator and the ability to identify invariants in the data.

A more careful description of how our method works, mathematically, in terms of producing convolutional filters and their respective effects, in terms of compressions, covariant and invariant transforms is presented in the Appendix.

3 Methods

Following (11), we use the Haar Scattering Network to decompose the original signal in a number of feature-signals, that represent data invariances and symmetries and we feed those features into regressors or classifiers, depending on the type of problem. This architecture is shown in Figure 2. However, in the present paper, we modified part of the architecture structure aiming to improve its performance. Instead of directly feeding the features into the regressor/classifier, for some classification problems, it is better to introduce a feature transformation layer, which is later fed into the regressor/classifier routines.

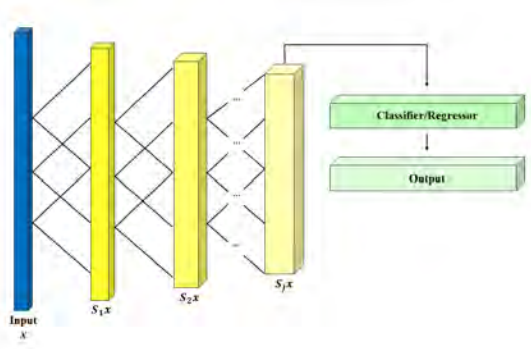


Figure 2: Graphical representation of the original algorithm described on (11).

The idea of introducing this transformation layer in the original architecture, which was not studied in (11) or in (8), is to bestow a simple pooling layer - in case of simpler signals - aiming to boost the dimensionality reduction; and to improve the separability of the features in a more robust dimensional space.

When only one property is calculated (such as max, min or $mean$), this transformation layer is a pooling layer. On the other hand, when statistical moments, autocorrelation and partial autocorrelation functions are calculated, data can be mapped into other feature spaces, which enhances the quantity of statistical / spectral information available.

While the first approach (simple pooling layer) seems to be an answer to the increasing number of features when the network becomes deeper - counterbalancing the number of features that are fed into the classifier; the second one (spectral features) is an alternative when simple features cannot be linearly separated within simple dimensional spaces by the classifier.

Also, following the ideas in (14), instead of using only the last layer S_j of the Haar Scattering Network, we observed that results may be significantly better when the inputs of a lower layer are made available to the transformation layer, resembling residual CNNs, of course, depending on the given problem. This is justified by eventual need of maintaining multiscale information about the signal, such as information contained in different frequencies/time scales - in case of time series.

That said, this connectivity increases even more the number of extracted features, pointing towards the necessity of introducing the aforementioned transformation layers: for pooling purposes - in the case of a large number of hidden layers; or for providing a more robust dimensional space, while maintaining relevant multiscale information.

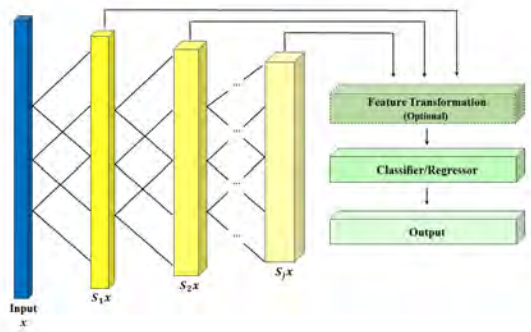


Figure 3: Graphical representation of the algorithm used in this work.

All these changes are summarized in Figure 3 and can be directly compared to the original architecture, which is summarized in Figure 2.

Aiming at a better understanding of the algorithm, the processing scheme defined in Figure 1 is written as pseudocode, as in Algorithms 1 and 2, where "TF" stands for "Transfer Function", which is the non-linear operator that should be applied over the differences inherent to the Haar wavelet, possibly different from the absolute value operator proposed in (11); σ, τ and L are respectively the scale, shift parameters of the scattering transform; and the number of layers.

Data: S_j, σ and τ;
Result: S_{j+1};
N = length(x);
S_{j+1} = NewEmptyLayer(N, j);
N' = N * 2^{-j};
for $1 <= n <= N'$ **do**
  for $1 <= k <= N'/2$ **do**
    $S_{j+1}[n, 2*(k\text{-}1)+1] = S_j[n, k] + S_j[2^{1-j} * N * \sigma + \tau + n, k]$;
  end
  for $1 <= k <= N'/2$ **do**
    $S_{j+1}[n, 2*k]$ = TF($S_j[n, k]$ - $S_j[2^{1-j} * N * \sigma + \tau + n, k]$);
  end
end

Algorithm 1: Internal Layer Processing - Function "HaarLayer"

Data: x, σ, τ and L;
Result: $S_j x$;
S_0 = inputLayer(x);
for $0 <= j <= L$ **do**
  S_{j+1} = HaarLayer(S_j, σ, τ);
end

Algorithm 2: General Haar Network Layer Processing - Function "HaarNetwork"

To obtain the code with residual channels (as shown in Figure 3), the implementation follows the same scheme, but, instead of returning only the last layer, one should create an auxiliary data structure to save the intermediate layers.

Estimations for regression problems are carried out as in Algorithm 3.

Data: x_t;
Result: Optimal σ, τ, R^2 and $\mathcal{R}$ (Regression Model);
$\sigma = \sigma_0$;
$\tau = \tau_0$;
while $|R^2_{k+1} - R^2_k| > \varepsilon$ **do**
  $F(x_t)$ = HaarNetwork(x_t, σ_k, τ_k);
  σ_{k+1}, τ_{k+1}, R^2_{k+1}, $\mathcal{R}$ =
  Optimize{Regression($F(x_t)$, x_t), σ_k, τ_k, L};
end

Algorithm 3: General Estimation Procedure for Regression Problems

Estimations for classification tasks are carried out as in Algorithm 4.

Predicting k steps ahead requires, in terms of regression tasks, that new values of $F(x_{t+k})$ should also be predicted. To accomplish such task, the proposed prediction method for extracted features is to take advantage that they are multiple stochastic processes that preserve different symmetries and invariances (which can be non-linear) from the original signal, forecast $F(x_t)$ up to $F(x_{t+k})$, by means of Fourier Series (eventually with trends, depending on the signal) and then feed into the estimated regression model $\mathcal{R}$.

On the other hand, classifying out-of-sample observations only requires extracting $\mathcal{F}(x^*)$, given a new sample x^*, and calculating the output from the estimated model $\mathcal{M}(\mathcal{F}(x^*))$.

Having proposed and described how the algorithm works, an assessment of its performance is needed, in order to compare the proposed method with other well established methods. To accomplish this goal we proceeded as follows.

For classification problems, we analyzed 18 datasets from the UEA & UCR Time Series Classification Repository (1). They are named as: “Computers", “Synthetic Control", “ECG - 200", “ECG - 5000", “Earthquakes", “Medical Images", “Phonemes", “FaceAll", “Mallat", “Distant Phalanx Age and Groups", “Fish", “Adiac", “Haptics", “Insect Wings", “BeetleFly", “FordA", “Chlorine Concentration" and “Inline Skate".

Data: x, ClassOf(x);
Result: Optimal σ, τ, C (Success Counts) and
$\mathcal{M}$ (Classification Model);
$\sigma = \sigma_0$;
$\tau = \tau_0$;
while $|C_{k+1} - C_k| > \varepsilon$ **do**
  $\mathcal{F}(x)$ = FeatureTransform(HaarNetwork(x, σ_k, τ_k));
  σ_{k+1}, τ_{k+1}, C_{k+1}, $\mathcal{M}$ =
  Optimize{Classification($\mathcal{F}(x)$, ClassOf(x)), σ_k, τ_k, L};
end

Algorithm 4: General Estimation Procedure for Classification Problems

For benchmarking purposes, in this kind of task, simple accuracy measures were used (percentage of correct classifications) to assess their performance in the test set. AUC and ROC measures were not calculated here, due to the fact that the original benchmarks were performed only using error in the test sets, as can be verified in (1). All the training and test sets were also implemented (i.e. divided) as provided in [1], in particular to have the same baseline for the results comparison.

Earlier studies, as the work of Bruna and Mallat [7] etc have confirmed the comparable state-of-the art performance of Deep Scattering Networks compared to CNNs on ImageNet and MNIST databases, we did not consider them herein, in addition to being out of scope to our unidimensional dataset problems study.

In terms of regression problems, we analyzed 5 very well known datasets in time series analysis: “Lung Cancer Deaths - UK", “Average Monthly Temperature - Nottingham", “Quarterly Gas Consumption - UK", “Monthly totals of international airline passengers" and “Mauna Loa Atmospheric CO2 Concentration". In these cases, out-of-sample R^2 measures were used to assess their performance.

The regression and classification algorithms that we used were Ordinary Least Squares (OLS) estimator, Random Forests (15), SVMs (10), Conditional Trees (5) and Recursive Partitioning (16). The whole implementation of the routines was made possible through the R Statistical Package (27). Instead of implementing regression and classification algorithms, $rpart$ (29), $ctree$ (17), $libsvm$ (25) and $randomForest$ (21) were respectively used for Recursive Partitioning, Conditional Trees, SVMs and Random Forests.

4 Results

Table 1 shows the results for regression and forecasting tasks, while in Table 2 are shown the results for classification tasks.

Table 1: Summary of the results obtained in the Regression/Forecasting tasks.

Dataset	Out-of-sample Observations	Haar Network R^2	ARIMA Model R^2	ETS Model R^2	Automatic Estimated ARIMA Model
Lung Cancer Deaths - UK	12	0.8920	0.7687	0.9509	ARIMA(2,0,1)
Average Monthly Temperature - Nottingham	24	0.9355	0.9243	0.9561	ARIMA(5,0,1)
Quarterly Gas Consumption - UK	12	0.9063	–	0.7197	ARIMA(2,1,3)
Monthly totals of international airline passengers	12	0.9360	0.9791	0.6363	ARIMA(4,1,3)
Mauna Loa Atmospheric CO_2 Concentration	24	0.8828	0.4326	0.2709	ARIMA(3,1,4)

In regression/forecasting problems, all results are expressed in terms of the R^2 measures, which were calculated using the whole out-of-sample set of observations against the predicted sets (test set), in order to verify the forecasting capabilities of the model as a whole, for each dataset, instead of assessing the capabilities for each observation.

Table 2: Summary of the results obtained in the Classification tasks.

Dataset	No. of Classes	Haar Network Accuracy	Best Model Accuracy	Optim. Procedure	Feature Transf. Layer	T.F.	No. of Layers	Residual Channels	Classif. Algorithm
Computers	2	0.7480	0.8	Nelder & Mead	Maximum Value	abs	4	Yes	Random Forest
Synthetic Control	6	0.9867	0.9992	Grid Search	Median Value	$tanh$	6	Yes	Random Forest
ECG 200	2	0.91	0.8905	Nelder & Mead	Median Value	$tanh$	4	No	SVM
ECG 5000	5	0.9155	0.9461	Grid Search	Mean Value	σ	5	No	Conditional Trees
Earthquakes	2	0.7410	0.7592	Nelder & Mead	Maximum Value	abs	2	Yes	Recursive Partitioning
Medical Images	10	0.7118	0.7850	Grid Search	Spectral Properties	abs	3	Yes	Random Forest
Phonemes*	39	0.3387	0.3620	Nelder & Mead	Spectral Properties	$tanh$	3	Yes	SVM
FaceAll	14	0.9448	0.99	Nelder & Mead	Spectral Properties	σ	3	Yes	SVM
Mallat	8	0.8899	0.9742	Grid Search	Maximum Value	abs	7	Yes	SVM
Distant Phalanx Age Groups	3	0.7480	0.8293	Grid Search	Spectral Properties	$tanh$	2	Yes	Random Forest
Fish	7	0.88	0.9742	Grid Search	Minimum Value	σ	5	No	SVM
Adiac	37	0.7775	0.8098	Grid Search	Spectral Properties	σ	2	Yes	SVM
Haptics	5	0.4870	0.5096	Nelder & Mead	Spectral Properties	$tanh$	3	Yes	SVM
Insect Wings	11	0.6389	0.6389	Grid Search	Maximum Value	σ	6	No	Random Forest
BeetleFly	2	0.9000	0.9485	Nelder & Mead	Maximum Value	σ	3	No	SVM
FordA	2	0.9076	0.9654	Nelder & Mead	Spectral Properties	$tanh$	3	Yes	SVM
Chlorine Concentration	3	0.8804	0.8457	Grid Search	Mean Value	$tanh$	7	Yes	SVM
Inline Skate*	7	0.6343	0.5525	Grid Search	Spectral Properties	abs	2	Yes	SVM

Moreover, in all these problems, the number of layers in the Haar network structure was fixed at 6; absolute value operator was chosen as non-linear operator following (11); all estimation procedures were carried out using the grid search method provided in (28); the input signals were interpolated using cubic-splines, to provide a larger amount of data to be processed, in order to increase the number of degrees of freedom; and the regression algorithm used was the Ordinary Least Squares (OLS) estimator.

It is also worth mentioning that no Feature Transformation layers were used in these problems, while the residual channels scheme, as in Figure 2, were used in all of them, in order to preserve multiscale information.

For benchmarking purposes, the results obtained using Haar Networks were compared to well known standard methods in time series analysis, as can be seen in (13): ARIMA (autoregressive integrated moving average) models and ETS (error, trend, seasonality) models. Aiming at the estimation of ARIMA models, lags selection was carried out using the $auto.arima$ procedure provided in the R statistical package - which detects the best ARIMA structure using statistical information criteria - for each time series. For the ETS models we used the ets procedure. Both procedures are provided in $forecast$ R package (18).

In Table 2, we show the results for each dataset, specifying: optimization procedure; the type of feature transformation layer; type of non-linear operator (transfer-function); number of Haar Network Layers (varied from 2 to 7); if residual channels are present; and, finally, the classification algorithm used.

In addition to that, it is important to notice that 5 different types of feature transformation layers were used: Maximum Value; Minimum Value; Median Value and Mean Value - these four acting as traditional pooling layers; and a different type that calculates some spectral properties of the extracted features. In this case: the first 4 values of the autocorrelation and partial autocorrelation functions - which characterize some of the spectral properties; plus the 4 first statistical moments: mean, variance, kurtosis and skewness.

Three different types of transfer functions (non-linear operators) were also tested: sigmoid function (denoted by σ); absolute value operator (denoted by abs) and hyperbolic tangent function (denoted by $tanh$). Also, in Table 2, dataset names marked with an asterisk indicate whether there is a rebalance in the training set and test set. This procedure was carried out because the original training sets were too small to train a SVM classifier.

5 Discussion

From the regression tasks perspective, which can be seen in Table 1, a key finding emerges: the performance of Haar networks in five different well-known datasets, in comparison to the well established methods such as ARIMA and ETS models, show that the proposed method is possibly more robust than its counterparts on the average, at least in this class of problems.

Despite not always providing the best performance, the proposed model had an out-of-sample R^2 measure above 88% in all datasets. On the other hand, the performance of ARIMA and ETS models varied in a wide range, from 27% to 97%. This highlights how stable the performance delivered by the proposed method is.

Future research should extend these tests and confirm, to what extent, this method is more robust than its counterparts.

That said, it is important to observe that the results of the experiments in regression tasks found clear support for the fact that the algorithm performs very well in the presence of symmetries and invariances (such as strong seasonal/periodic components) in the data, given that all these time series have linear and non-linear cyclical components.

It is worth noticing that these interesting facts are in line with the perspective of understanding how the proposed algorithm works, in terms of feature extraction, which basically operates by decomposing time series / signals into feature sets that preserve their symmetries.

From the perspective of classification tasks, which can be seen in Table 2, our results cast a new light on how invariances and symmetries play a fundamental role in 1D signal classification.

First, it is important to highlight that, for some of these problems, we observed that changing the non-linear transfer function from the original absolute value operator as seen in (11), to others, such as $tanh(t)$, made the algorithm perform better.

Second, we are able to observe empirically, as a clear trade-off, that simple transformation layers are used (to account for dimensionality reduction) for deeper networks (number of layers greater than 3); while, for shallower networks, more complex transformation layers are needed.

Superior results are seen for “ECG 200", “Chlorine Concentration" & “Inline Skate" datasets, while a negligible improvement is seen in “Insect Wings" dataset. On the remaining datasets, on average, our proposed algorithm is outperformed by 10% - in terms of the relative performance - by the best algorithms, as compiled by (1).

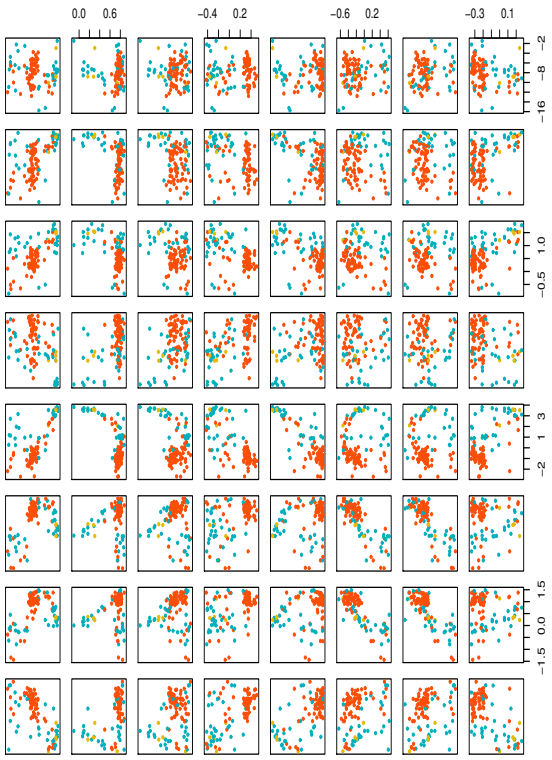
Keeping in mind that the proposed algorithm relies on extracting invariances and symmetries and feeding them into an external classifier, this analysis found evidence that this kind of feature plays a fundamental role in 1D signal classification and that a further understanding in conjunction with other well established concepts, such as dynamic time warp, is needed.

It is also worth mentioning that spectral and multiscale features of time-series data can represent some important behavior of the system that are not obvious in the time-series domain. It is possible, for example, to use spectral characteristics such as the data’s frequency and power domain, to extract signal periodicities and reduce data noise.

As already explained before, our approach, an adaptation of (11) and (9), feeds these spectral and multiscale features to a regression / classification algorithm, in order to construct a model of the processes based solely on the sampled data, being an interesting alternative to ARIMA models and to traditional artificial neural networks (ANNs), in a way that additional insights can be retrieved in comparison to these traditional methods.

As can be seen in Figures 4, 5 and 6, our method clearly circumvents one of the major drawbacks of ANNs (and even CNNs), which is the fact that they are usually considered "black-boxes", meaning that it is difficult to understand why the algorithm makes a particular decision (4).

In the first case (Fig. 4) it is possible to verify that, naturally, when appropriate transformations are used, clusters arise in different dimensions, allowing desired classification properties. In this case, how myocardial infarction occur, based on the clustering of the features. Dots in red represent normal conditions, turquoise dots are myocardial infarctions and yellow dots represent misclassified cases.



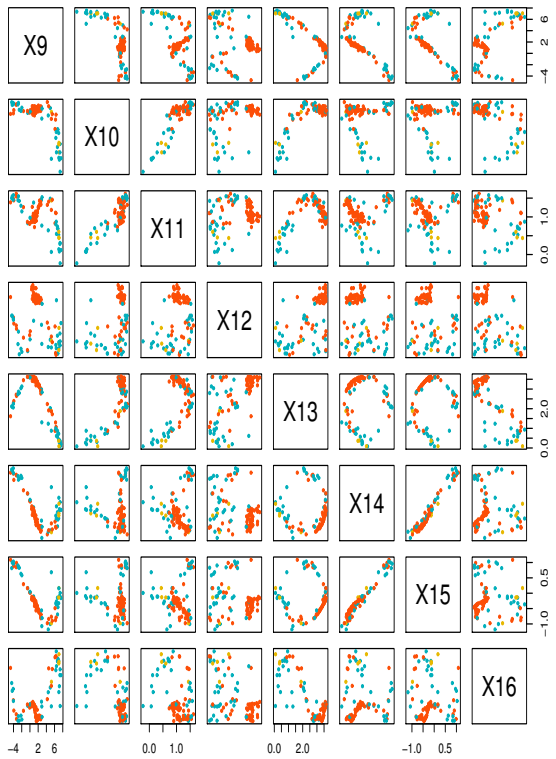


Figure 4: Clustering of features of myocardial infarction in different dimensions.

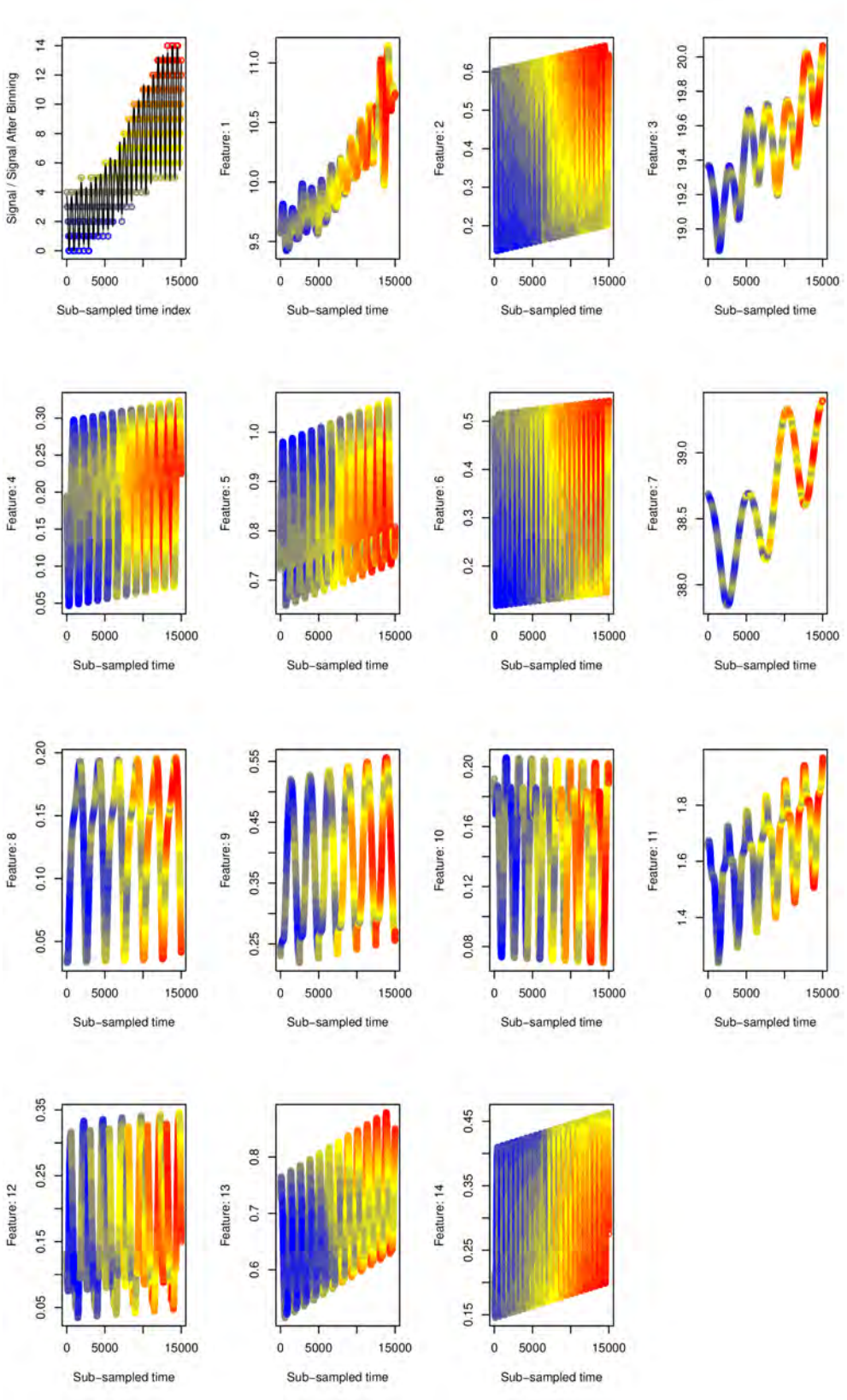


Figure 5: Time Series and its extracted features - UK Gas Consumption Time Series.

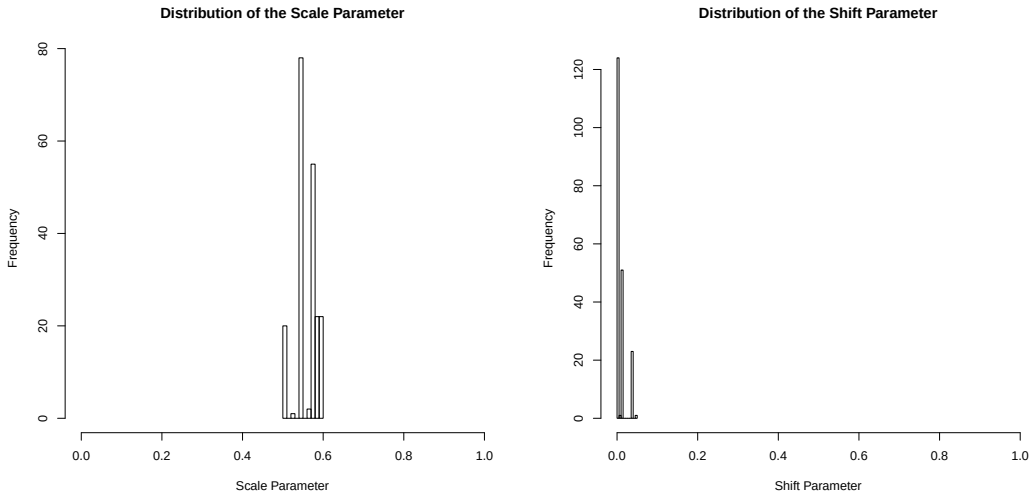


Figure 6: Distribution of the scale and shift parameter under training set resampling - ECG-200.

In the second case (Fig. 5), it is possible to observe that, if different levels in the original signal are assigned to a specific color set (palette), in this case resembling the colors of a heatmap-like graphic, it is possible to understand how different multiscale properties and their respective signs are linked to the composition of the original signal levels, in such a way that colors close to "red" represent the observations with higher values, and the opposite as they get closer to blue, providing straightforward information on how these features are correlated to the original process in terms of cycles, trends and irregular components - which is something that ARIMA models and ANN-based models usually do not provide.

In Fig. 6, we verify the stability of the parameters when samples are randomly reshuffled. This is one of the major features of the proposed architecture. As all convolutional filters are directly linked to scale (σ), shift (τ) and number of layers L, it is possible to apply straightforward comparisons, in a low-dimensional manner, between different estimations - something that is much harder to accomplish on other more complex architectures such as ANNs. This is a desirable feature especially when dealing with time series problems where structural breaks or regime switches may occur.

6 Conclusions

This paper proposes an extension of the architecture provided in (11) - which is basically an adaptation of the architecture developed in (9) and in (23), focusing on 1D signals classification and regression tasks of pattern recognition. This architecture, basically consists of stacking convolutional filters, that can be thought as a generalization of Haar wavelets (where we demonstrate such statement in the Appendix), followed by non-linear operators, which aim to extract symmetries and invariances that are later fed in a classification/regression algorithm.

We obtained good results with this simple method, in a wide range of datasets, for both kind of tasks. Furthermore, despite the fact that dataset descriptions can be found in (2), it is important to highlight and emphasize their important potential real-life applications, such as Industrial Control Charts ("Synthetic Control"); Myocardial Infarction detection ("ECG - 200"); detection of disease-vectors ("Insect Wings"), such as Aedis Egypt and water quality ("Chlorine Concentration"). Medical applications based on using such a computationally efficient and simpler architecture is also suited to applications such as wearable health devices (31). Thus, the impact of this line of research in pattern recognition is potentially considerable.

The same conclusions can be drawn for regressions in the presence of invariances and symmetries, which are interesting to a broader audience of professionals who seek massive automatic modeling, without the need of further investigations, such as economists, engineers and data scientists.

As pointed out in the introductory section, the results may also pave the way to the development of new AutoML algorithms, providing functional models with minimal human intervention, given its potential of generalization.

In addition to that, it is important to emphasize the fact that all convolutional structures can be explained and mapped just on top of only three parameters, as shown here. Hence, this proposed architecture also enables further analysis, such as direct comparison between multiple convolutional structures, allowing one to evaluate whether a structure is stable over time, or what are the effects that resampling cause in the whole convolutional structure - which is somewhat difficult to be done in the traditional CNN/MLP framework. The direct comparison between these three hyperparameters allows one to draw straightforward conclusions about how different are two or more convolutional structures.

In the context of traditional CNN/MLP, all parameters are estimated by applying a (stochastic) gradient descent algorithm in conjunction with a backpropagation algorithm, which (theoretically) enhances the forecasting/classification capabilities, but also bring a lot of complexity in terms of their respective analysis.

That said, given the considerable flexibility to adapt the architecture to different problems, by means of modifying the non-linear operator, number of layers, feature transformation layer and the classifier itself, it is imperative to extend this research, in further works, to enable a better comprehension of how much can be improved, since the search for the best architecture was primarily hand-made. Other classification algorithms and non-linear operators, which were not considered here, can also be included.

References

[1] Bagnall, A., Lines, J., Bostrom, A., Large, J., Keogh, E.: The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. Data Mining and Knowledge Discovery **Online First** (2016)

[2] Bagnall, A., Lines, J., Bostrom, A., Large, J., Keogh, E.: The UEA & UCR time series classification repository (2016). Http://timeseriesclassification.com/

[3] Bagnall, A., Lines, J., Hills, J., Bostrom, A.: Time-series classification with cote: The collective of transformation-based ensembles. IEEE Transactions on Knowledge and Data Engineering **27** (2015)

[4] Benitez, J., Castro, J., Requena, I.: Are artificial neural networks black boxes? IEEE Transactions on Neural Networks (1997)

[5] Breiman, L., Friedman, J., Olshen, R., Stone, C.: Classification and Regression Trees (1984)

[6] Bronstein, M.M., Bruna, J., LeCun, Y., Szlam, A., Vandergheynst, P.: Geometric deep learning: going beyond euclidean data. IEEE Signal Processing Magazine (2017)

[7] Bruna, J., Mallat, S.: Invariant scattering convolution networks. IEEE Transactions on pattern analysis and machine intelligence **35**(8) (2013)

[8] Cheng, X., Chen, X., Mallat, S.: Unsupervised deep haar scattering on graphs (2014). ArXiv:1406.2390

[9] Cheng, X., Chen, X., Mallat, S.: Deep haar scattering networks. Information and Inference: A Journal of the IMA **5** (2016)

[10] Cortes, C., Vapnik, V.: Support vector networks. Machine Learning **20**, 273–297 (1995)

[11] Fernandes, F.N.: Building function approximators on top of haar scattering networks (2018). ArXiv:1804.03236v1

[12] Fino, B.J.: Relations between haar and walsh/ hadamard transforms **60**(5), 647–648 (1972). ArXiv:1101. 2286

[13] Hamilton, J.D.: Time Series Analysis. Princeton University Press (1994)

[14] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. CoRR **abs/1512.03385** (2015). URL `http://arxiv.org/abs/1512.03385`

[15] Ho, T.K.: Random decision forests pp. 278– (1995). URL `http://dl.acm.org/citation.cfm?id=844379.844681`

[16] Hothorn, T., Hornik, K., Zeileis, A.: Unbiased recursive partitioning: A conditional inference framework. Journal of Computational and Graphical Statistics **15**(3), 651–674 (2006)

[17] Hothorn, T., Hornik, K., Zeileis, A.: ctree: Conditional Inference Trees (2018). URL `https://cran.r-project.org/web/packages/partykit/vignettes/ctree.pdf`. 'R' package version 1.2-2

[18] Hyndman, R.: Support Vector Machines: The Interface to libsvm in package e1071 (2018). URL https://cran.r-project.org/web/packages/forecast/forecast.pdf. 'R' package version 8.4

[19] Karlsson, L., Lungkvist, M., Loutfi, A.: A review of unsupervised feature learning and deep learning for time-series modeling. Pattern Recognition Letters **42** (2014)

[20] Lecun, Y., Kavukcuoglu, K., Farabet, C.: Convolutional networks and applications in vision pp. 253–256 (2010)

[21] Liaw, A.: Package 'randomForest' (2018). URL https://cran.r-project.org/web/packages/randomForest/randomForest.pdf. 'R' package version 4.6-14

[22] Mallat, S.: A Wavelet Tour of Signal Processing, The Sparse Way. Academic Press (2009)

[23] Mallat, S.: Group invariant scattering (2011). ArXiv:1101. 2286

[24] Mallat, S.: Understanding deep convolutional networks . Physical Transactions A **374** (2016)

[25] Meyer, D.: Support Vector Machines: The Interface to libsvm in package e1071 (2017). URL https://cran.r-project.org/web/packages/e1071/vignettes/svmdoc.pdf. 'R' package version 1.6-8

[26] Nelder, J.A., Mead, R.: A simplex method for function minimization. Computer Journal **7**(4), 308–313 (1965). DOI 10.1093/comjnl/7.4.308

[27] R Core Team: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria (2013). URL http://www.R-project.org/

[28] Schuhmann, E.: Package 'NMOF' (2018). URL https://cran.r-project.org/web/packages/NMOF/NMOF.pdf. 'R' package version 1.4-3

[29] Therneau, T., Atkinson, B., Ripley, R.: Package 'rpart' (2018). URL https://cran.r-project.org/web/packages/rpart/rpart.pdf. 'R' package version 4.1.13

[30] Wong, C., Houlsby, N., Lu, Y., Gesmundo, A.: Transfer automatic machine learning (2018). ArXiv:1803.02780

[31] Zhang, H., Sun, Y., Lu, Y., Lan, J., Ji, Y.: A novel motion and noise artifacts reduction mechanism (mnarm) for wearable ppg-based heart rate extraction pp. 1–4 (2015). DOI 10.1049/cp.2015.0785

7 Appendix: Haar Scattering Networks as Generic Convolutional Filters

First, it is important to remind that, for a typical Haar Wavelet, the filter is given by:

$$H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{A.10}$$

$$H_4 = \begin{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \bigotimes [1 \quad 1] \\ \mathrm{I}_2 \bigotimes [1 \quad -1] \end{bmatrix} \tag{A.11}$$

$$\dots$$

$$H_N = \begin{bmatrix} \mathrm{H}_{N/2} \bigotimes [1 \quad 1] \\ \mathrm{I}_{N/2} \bigotimes [1 \quad -1] \end{bmatrix} \tag{A.12}$$

For Haar Scattering, we may describe the Transform as:

$$W = f(HST_N \cdot X) \tag{A.13}$$

where f denotes a non-linear transfer function, HST_N denotes the Matrix of the Transform (i.e. the convolutional matrix that is applied before the non-linear function), X the input vector, and W the resulting vector.

Furthermore, we can describe the design matrix of the transform, see Section 2.1, where $\bigotimes$ is the Kronecker product (also known as the Hadamard product) as:

$$HST_N = \begin{bmatrix} \mathrm{A}_{P,\sigma,\tau} \bigotimes \begin{bmatrix} 1 \\ 1 \end{bmatrix} & \mathrm{B}_{P,\sigma,\tau} \bigotimes \begin{bmatrix} 1 \\ -1 \end{bmatrix} \end{bmatrix} \tag{A.14}$$

where $A_{P,\sigma,\tau}$ and $B_{P,\sigma,\tau}$ are permutations of the Identity Matrix, shifted according to σ and τ, and $P = log(N)/log(2)$.

If $\sigma = 1$ and $\tau = 0$, it follows that:

$$HST_{N=2} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{A.15}$$

$$HST_{N=4} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & -1 \end{bmatrix} \tag{A.16}$$

$$HST_{N=8} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 \end{bmatrix} \tag{A.17}$$

If $\sigma = 1$ and $\tau = 1$, it follows that:

$$HST_{N=2} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix} \tag{A.18}$$

$$HST_{N=4} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{A.19}$$

$$HST_{N=8} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{A.20}$$

That said, it is possible to show that for $P = \frac{log(N)}{log(2)}$, HST_N^P is a Hadamard Matrix, for the following specific choices: $\sigma = 1$ and $\tau = 0$. We show this as below.

We recall the following property of Kronecker product:

$$\big(A \bigotimes B\big) \cdot \big(C \bigotimes D\big) = \big(AC\big) \bigotimes \big(BD\big) \tag{A.21}$$

where A and C are matrices of the same column and row size respectively, and similarly for B and D.

Furthermore, knowing that:

$$HST_N^P = HST_N^{P-1} \cdot HST_N \tag{A.22}$$

And given the definition of the Hadamard Matrices (H_p):

$$H_p = H_{p-1} \bigotimes H_1 \tag{A.23}$$

where H_1 is equivalent to HST_1, H_p becomes

$$H_p = \begin{bmatrix} \mathrm{H}_{p-1} & \mathrm{H}_{p-1} \\ \mathrm{H}_{p-1} & -\mathrm{H}_{p-1} \end{bmatrix} \tag{A.24}$$

If we choose $\sigma = 1$ and $\tau = 0$, it is convenient to write (A.14) relationship as :

$$HST_N^P = \begin{bmatrix} \mathrm{I}_P \otimes \begin{bmatrix} 1 \\ 1 \end{bmatrix} & \mathrm{I}_P \otimes \begin{bmatrix} 1 \\ -1 \end{bmatrix} \end{bmatrix} \tag{A.25}$$

It follows from (A.22) that:

$$\begin{aligned} HST_N^2 = \begin{bmatrix} I_P \otimes \begin{bmatrix} 1 \\ 1 \end{bmatrix} & I_P \otimes \begin{bmatrix} 1 \\ -1 \end{bmatrix} \end{bmatrix} \cdot \\ \begin{bmatrix} I_P \otimes \begin{bmatrix} 1 \\ 1 \end{bmatrix} & I_P \otimes \begin{bmatrix} 1 \\ -1 \end{bmatrix} \end{bmatrix} \end{aligned} \tag{A.26}$$

$$\begin{aligned} HST_N^3 = \left(\begin{bmatrix} I_P \otimes \begin{bmatrix} 1 \\ 1 \end{bmatrix} & I_P \otimes \begin{bmatrix} 1 \\ -1 \end{bmatrix} \end{bmatrix} \right)^2 \cdot \\ \begin{bmatrix} I_P \otimes \begin{bmatrix} 1 \\ 1 \end{bmatrix} & I_P \otimes \begin{bmatrix} 1 \\ -1 \end{bmatrix} \end{bmatrix} \end{aligned} \tag{A.27}$$

$$\begin{aligned} HST_N^P = \left(\begin{bmatrix} I_P \otimes \begin{bmatrix} 1 \\ 1 \end{bmatrix} & I_P \otimes \begin{bmatrix} 1 \\ -1 \end{bmatrix} \end{bmatrix} \right)^{P-1} \cdot \\ \begin{bmatrix} I_P \otimes \begin{bmatrix} 1 \\ 1 \end{bmatrix} & I_P \otimes \begin{bmatrix} 1 \\ -1 \end{bmatrix} \end{bmatrix} \end{aligned} \tag{A.28}$$

If P = 1, it follows that:

$$HST_{N=2} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = H_1 \tag{A.29}$$

By using the principle of mathematical induction, this generalizes to $H_P = (HST_N)^P$ (as proven in (12)). To verify that (A.24) still holds, let's express H_P explicitly as follows:

$$H_P = (HST_{N=2^{P-1}})^{P-1} \bigotimes H_1 \tag{A.30}$$

$$= \left(\begin{bmatrix} \mathrm{I}_P \otimes \begin{bmatrix} 1 \\ 1 \end{bmatrix} & \mathrm{I}_P \otimes \begin{bmatrix} 1 \\ -1 \end{bmatrix} \end{bmatrix} \right)^{P-1} \bigotimes H_1 \tag{A.31}$$

which reduces to

$$= \begin{pmatrix} \left(\begin{bmatrix} I_{P-1} \bigotimes \begin{bmatrix} 1 \\ 1 \end{bmatrix} & I_{P-1} \bigotimes \begin{bmatrix} 1 \\ -1 \end{bmatrix} \end{bmatrix} \right)^{P-1} & \left(\begin{bmatrix} I_{P-1} \bigotimes \begin{bmatrix} 1 \\ 1 \end{bmatrix} & I_{P-1} \bigotimes \begin{bmatrix} 1 \\ -1 \end{bmatrix} \end{bmatrix} \right)^{P-1} \\ \left(\begin{bmatrix} I_{P-1} \bigotimes \begin{bmatrix} 1 \\ 1 \end{bmatrix} & I_{P-1} \bigotimes \begin{bmatrix} 1 \\ -1 \end{bmatrix} \end{bmatrix} \right)^{P-1} & -\left(\begin{bmatrix} I_{P-1} \bigotimes \begin{bmatrix} 1 \\ 1 \end{bmatrix} & I_{P-1} \bigotimes \begin{bmatrix} 1 \\ -1 \end{bmatrix} \end{bmatrix} \right)^{P-1} \end{pmatrix}. \tag{A.32}$$

Where, by induction, it is possible to verify that $H_p = \begin{bmatrix} \mathrm{H}_{p-1} & \mathrm{H}_{p-1} \\ \mathrm{H}_{p-1} & -\mathrm{H}_{p-1} \end{bmatrix}$ still holds.

We illustrate this matrix representation of H_P in the following example for H_3:

$$H_3 = \begin{pmatrix} \left(\begin{bmatrix} I_2 \bigotimes \begin{bmatrix} 1 \\ 1 \end{bmatrix} & I_2 \bigotimes \begin{bmatrix} 1 \\ -1 \end{bmatrix} \end{bmatrix} \right)^2 & \left(\begin{bmatrix} I_2 \bigotimes \begin{bmatrix} 1 \\ 1 \end{bmatrix} & I_2 \bigotimes \begin{bmatrix} 1 \\ -1 \end{bmatrix} \end{bmatrix} \right)^2 \\ \left(\begin{bmatrix} I_2 \bigotimes \begin{bmatrix} 1 \\ 1 \end{bmatrix} & I_2 \bigotimes \begin{bmatrix} 1 \\ -1 \end{bmatrix} \end{bmatrix} \right)^2 & -\left(\begin{bmatrix} I_2 \bigotimes \begin{bmatrix} 1 \\ 1 \end{bmatrix} & I_2 \bigotimes \begin{bmatrix} 1 \\ -1 \end{bmatrix} \end{bmatrix} \right)^2 \end{pmatrix} \tag{A.33}$$

$$= \begin{pmatrix} \left(\begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & -1 \end{bmatrix} \right)^2 & \left(\begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & -1 \end{bmatrix} \right)^2 \\ \left(\begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & -1 \end{bmatrix} \right)^2 & -\left(\begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & -1 \end{bmatrix} \right)^2 \end{pmatrix} \tag{A.34}$$

$$= \begin{pmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} & \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \\ \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} & -\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \end{pmatrix} \tag{A.35}$$

which is the Hadamard matrix for a vector of size 8.

In general, it is possible to obtain a very large set of different possible convolutional filters, including traditional Haar, Walsh/Hadamard filters, on top of only three hyperparameters – σ (scale), τ (shift) and P number of stacked filters.

In addition to that, a fourth hyperparameter can be included, in conjunction with the transfer function that should be applied after the filtering, as described in Equation (9). For example, if this function $f(x)$ is a sigmoid or a hyperbolic tangent, a normalization factor (a fourth hyperparameter) λ can be included, in such a way that:

$$\lambda \cdot y = x \tag{A.36}$$

where linearities or non-linearities may prevail according to λ. To understand such statement, one must remember that:

$$||x|| << 1 \rightarrow f(x) \approx x \tag{A.37}$$

if $f(x)$ is similar to a hyperbolic tangent or sigmoid function. Keeping that in mind, in order to determine to what extent the filter itself becomes invariant or covariant, one can enhance the set of possible data transformations that can be made.

Thus, a high number of different data transformations can be achieved, by just modifying a very small set of hyper-parameters, providing an interesting toolset that can be easily optimized using traditional techniques such as grid search.