Experimental signal-noise ratio enhancing of magnetotelluric time series with machine learning

Author: Ing. José R. Campos,
Supervisor: Prof. PhD. Alex Marcuello.

Abstract.

Keywords: Magnetotelluric, Machine Learning, noise, LSTM, RNN, Stockwell S–transform

The Magnetotelluric Method (MT) has increasingly become important in reservoir characterization. Quality data has to be used to obtain effective results of the method; however it is usually registered with the addition of cultural noise. To improve the signal to noise ratio Machine Learning models can provide a useful approach taking advantages of the physical relationship between the electromagnetic (EM) components. A methodology to enhance the signal to noise ratio is established in this work. A specific type of Recurrent Neural Network (RRN), namely: Long Short-Term Memory Networks (LSTM) is proposed to recover the fundamental frequencies of the signal out of the noisy data. Feature engineering for the Machine Learning Model is done with the Stockwell Transform (S-transform) to work in the time-frequency domain. The methodology was applied first on synthetic data proving that a quality time-frequency representation (TFR) of the original signal (without added noise) can be recovered. Moreover, the methodology has been applied to real data obtaining a TFR of the signal fundamental frequencies. The established workflow can be further applied for similar time-series analysis and variations of it can be applied to similar geo-data problems.

1. Introduction and Objectives

In this work a supervised machine learning (ML) models will be utilized to improve the signal-to-noise ratio of electromagnetic (EM) time series data. The main reason why this is of relevance for the geophysics community is that the electromagnetic methods highly depend on the quality of the data registered, but nowadays it is uncommon to operate in noise-free environments (Vozoff K., 1991; Chave, A. D., & Jones, A. G., 2012).

As Dehghani, (2011), states “electromagnetic methods are now being used to provide images of subsurface resistance on the reservoir scale”. As long as the interest and convenience for the use of these methods keeps increasing, in particular the magnetotelluric method (MT), new techniques for improving the signal-to-noise ratio will gain more value. Providing better quality in the processing of the MT data and, therefore, better results for reservoir characterization.

To achieve the data analysis and treating, state-of-the-art methodologies will be put to test, specifically machine learning algorithms through the use of open source Python libraries mainly TensorFlow and Keras; both have been gaining popularity lately for the efficiency with which they are solving problems in diverse industries (Chollet, F., 2017).

1.1 Previous Research

Extensive research has been performed about the noise treatment for magnetotelluric data, such as those from Fontes et al. (1988); Junge (1996); Banks (1998); Oettinger et al. (2001); Weckmann et al. (2005); Escalas et al.(2013); Carbonari et al. (2017) among others.

In a general overview, these papers debate and/or concur in their conclusions that MT methods are, although effective, highly dependent on a good noise treatment, beyond that, various methodologies are tested to make improvements on the issue, being one of the latest tendencies to work with representations that allow to operate in the frequency domain without losing resolution on the time scale.

Regarding the use of ML as a method for data analysis and treatment in MT geophysics, recent research is available, mainly in the field of neural networks, among others: Bishop (1995); Cottrell et al. (1995); Calderón-Macías et al. (2000); Van der Baan and Jutten (2000); Manoj and Nagarajan (2003); Popova and Ogawa (2007); Bielecka et al.(2009). In a wide aspect, these works agree that ML has been efficient for the researches for which it has been tested.

1.2 Hypotheses

1. As well as in previously mentioned case-studies ML can be proven a useful methodology for the interest of signal-noise ratio enhancing.
2. MT time series correlation between the electric and magnetic fields is expected to be useful in the process of using ML to filter the noise in the data.
3. Using the S-transform as to pre-process the MT time series data for the ML algorithms may be an effective way to achieve favorable results.

1.3 Objectives

1. To create a workflow that can be followed and improved-on as a new methodology for treating MT noise.
2. To use up-to-date open source Python libraries for the noise-to-signal ratio enhancing of Mt time series.
3. To implement the S-transform as the time-frequency representation (TFR) of the time series data to be worked with the ML model.

1.4 Theoretical Background

Deep Learning is a subfield of Artificial Intelligence in computer science, enclosed in the Machine Learning field. It has become popular in the recent years given its advances in several fields; most relevant to this thesis are the advances in time series analysis known in the machine learning theories as sequential data analysis. This analysis is fundamental for most geophysical methods, mainly for the processing of time series data to achieve structure of the subsurface. A perfect example is Magnetotelluric processing, where data, usually consists of

four time series, sampled at known frequencies, corresponding to the horizontal electromagnetic field components. MT data is brought through physical principles to obtain the distribution of the ground electrical resistivity.

Derived from the Maxwell's equations (Cagniard 1953), these physical relations among the magnetic and electric field components are the fundamental basis of the MT method. Data work in the frequency domain and the relevant magnitude for the interpretation is the impedance, Z, defined by:

$$E = Z \cdot B \qquad \text{Eq. 1}$$

Where $\boldsymbol{E}$ and $\boldsymbol{B}$ are the EM horizontal fields at the surface in the frequency domain and $\boldsymbol{Z}$ is the impedance matrix that describes the medium. From the impedance other responses of the model like the apparent resistivity o phases curves can be calculated.

In classical MT processing this matrix Z is determined as the answer of an inverse problem (Parker, R. L. 1983; Chave, A. D., & Jones, A. G., 2012) by hardcoding the input rules (behavior of the medium) and providing the input data (EM field components). This is shown as classical programming in Figure 1, which also shows the proposed approach with machine learning. ML uses the data (e.g. Magnetic field components/variations) and the answer expected (e.g. Electric field components/variations) as an input, and the ML model uncovers the rules (behavior of the medium).

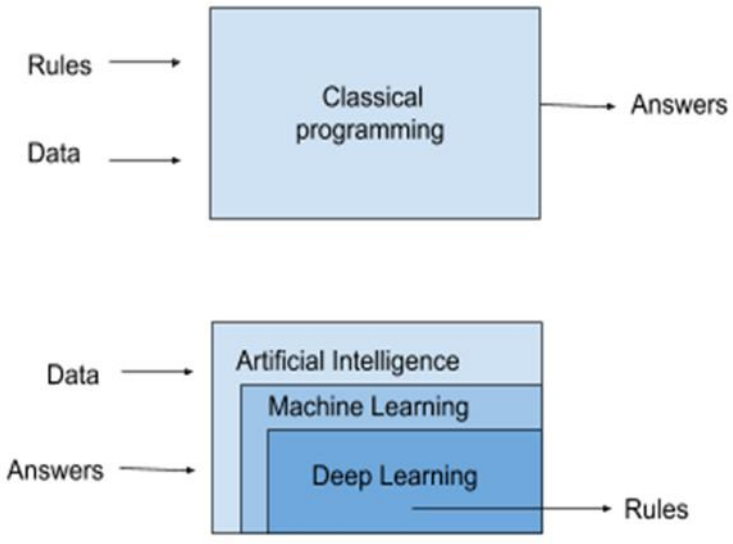

Figure 1 Comparison between classical programming and Machine Learning approach

The processing of the given input data and answer is known as training the model also fitting the model or learning (input data is also called training data). this process will be explained later. A better intuition to understand ML is that it finds a mapping function between the input data (training data) as shown in Figure 2.

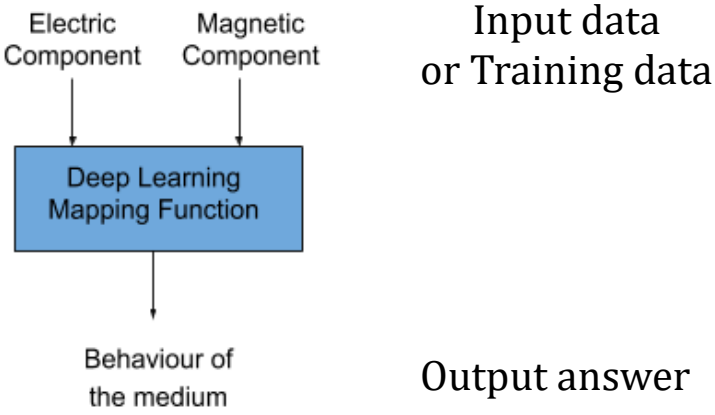

Figure 2 Proposed Machine Learning Model

In the particular case of the MT method we have an ideal setting, where physical principles allow for the assumption that there is a mapping function (MF) between the Magnetic field and Electric field. One of the hypotheses of this work is that ML models can outperform hardcoding by taking advantage from the aforementioned empirical relationship between the E and the B.

The following section will explain the process of training a recurrent neural network to find an MF between EM data to be later related to the impedance tensor.

1.4.1 Machine Learning Models

Machine Learning Models are plenty, and an extensive introduction on this topic would be out of the scope of this work. Artificial neural networks (ANNs) are a subdivision of machine learning algorithms that are architected with an interconnected group of nodes packed in layers; every node performs an independent optimization problem taking as input results from previous layers (see e.g., Goodfellow 2016). To work with sequential data (e.g. time series) and specific type of ANNs provides better results, namely: recurrent neural networks RNN. In particular, LSTM networks have achieved top results in this manner. (Graves et al. 2013; Wu et. al. 2016)

A very simplified explanation can be the following: neural networks learn by recursively (high number of iterations) minimizing a score obtained from a named Loss Function, which calculates the difference between the expected or real results (measured electric component) and the predicted results (predicted electric component). Predicted results are obtained from the mapping function that takes as input the magnetic component. The MF defines the electric component as a function of the magnetic component by

$$E(f) = W \cdot B(f) \qquad \text{Eq. 2}$$

where W is the matrix of weights and E & B the electromagnetic components in the time-frequency domain.

Minimizing the score of the Loss Function is done by the implementation of the gradient descent algorithm, an optimization method to find global minima. Gradient descent works by multiplying the weights times a learning rate opposite to the gradient. The backpropagation method is used to find the gradient of the Loss Function by applying the calculus chain rule. (see e.g., Werbo 1990). Figure 3 diagrams the previous described processes for a training iteration.

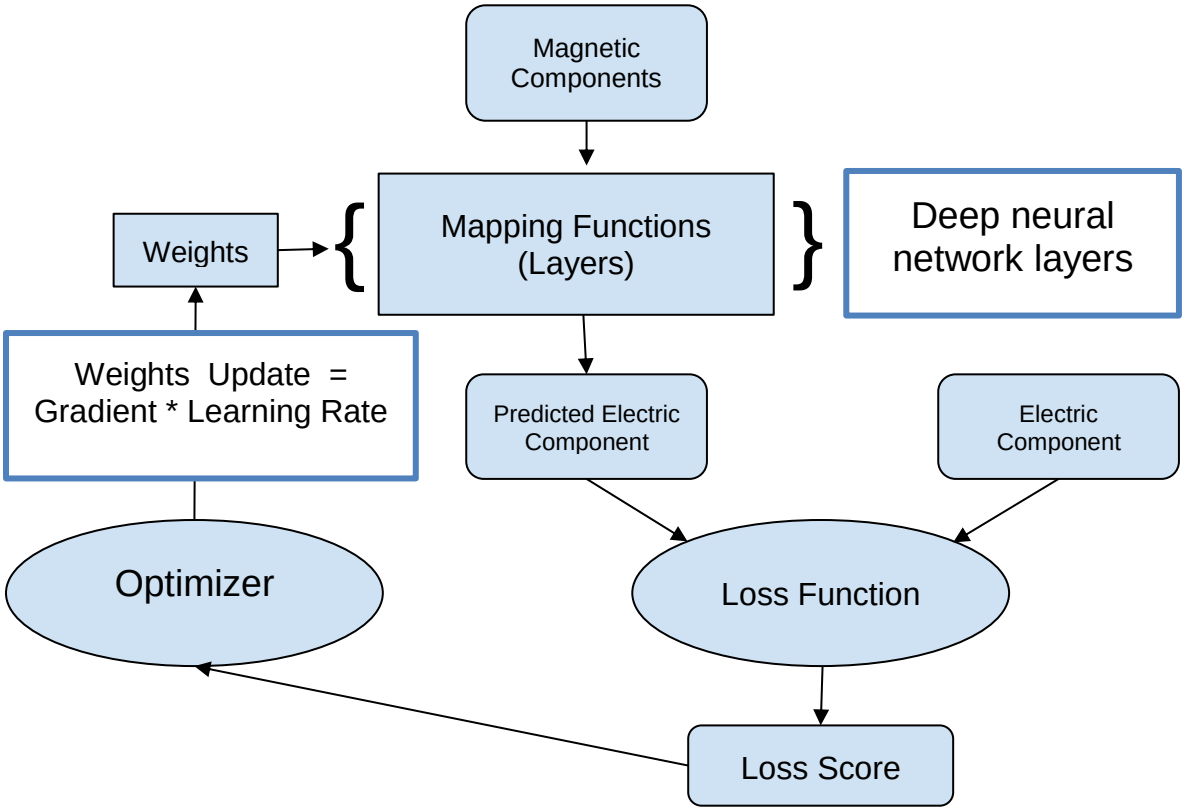

Figure 3. Machine Learning Model Training for one iteration

1.4.1.1 Recurrent Neural Networks (RNN)

Recurrent neural networks or RNNs (Rumelhart et al., 1986a) characterize for mapping every element in a sequence iteratively. For each of the iterations it stores information that is used along the following element in the next step.

In this work, at each step t analyses a time batch of the specified magnetic component in the time-frequency domain ($x^{\langle t \rangle}$), then the information is used to optimize the mapping function that leads to the electric component for the correspondent time batch ($\hat{y}^{\langle t \rangle}$).

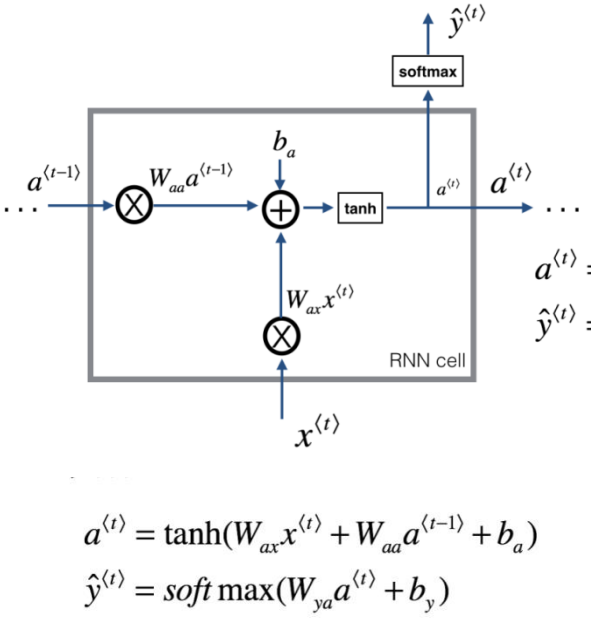

Figure 4. Basic algorithm of the Recurrent Neural Networks, modified from Ng, A. (2018).

Figure 4 depicts the basic algorithm of a RNN at an specific time (t) step. Where $a^{\langle t \rangle}$ is the activation function calculated as the tangent of a linear equation given by the cell weights W_{ax} times $x^{\langle t \rangle}$ plus the activations weights W_{aa} times the previous activation values $a^{\langle t-1 \rangle}$ (stored/cached information) plus a bias b_y. And $\hat{y}^{\langle t \rangle}$ equals to sigmoid function of a linear transformation of the activation function.

Naturally RNN are the ideal approach to process sequential data, but when mapping long term dependent features (such as occasional cultural noise) a problem arises, known as the "vanishing gradient" (Hochreiter 1991; Bengio et al., 1993, 1994). It is usually expected in deep RNN where the backpropagation algorithm tends to collapse.

1.4.1.1.1 LSTM Networks

The Long Short-Term Memory (LSTM) algorithm was developed by Hochreiter and Schmidhuber in 1997. It was the culmination of their research on the vanishing gradient problem, the solution was to design an algorithm ("cell") where logistic and linear units ("gates") interacted at each time step to allow information to be reused at a later time steps. The model presented in this thesis is based on LSTM networks.

1.4.2 Machine Learning Workflow

Typically in a Machine Learning Model implementation the available data set is divided in two sub-sets, the training and the validation data. The training data is used as explained before to find the mapping function and the validation set of data is used to verify the accuracy of the prediction obtained with that mapping function.

In this particular case (1D homogenous half-space) it is possible to use one component of the magnetic field (Bx) with its corresponding component of the electric field (Ey) as a training data to find the mapping function. Given that this mapping function should work as well with the left components, the validation data will be constituted of By and Ex as shown in the following equations.

$$Ey = MappingFunction(Bx, By) \qquad Training\ Model$$
$$Ex = MappingFunction(Bx, By) \qquad Validation\ Model$$

Recall that these equations are in the frequency domain, and the actual data is in the time domain. Therefore a pre-processing of the data must be done, in ML terms this is known as Feature Engineering.

1.4.2.1 Feature Engineer

Feature engineering is the pre-processing of the input data aiming to obtain better mapping functions from the machine learning model. In this work we provide the input data in the time-frequency domain to the ML model, therefore the MF in this domain; hence it can be directly related to the impedance tensor Z.

A high resolution transformation of the EM components to the frequency domain is essential to produce a reliable mapping function, this can be done with the short-time Fourier transform (STFT), but the time-frequency representation (TFR) obtained from the STFT fails to provide competent resolution when the signal is composed of a wide range of frequency spectrum, which is the case of MT components. For this reason transformations based on Wavelet (WL) transforms provide a better approach to the limited resolution of the SFTF.

1.4.3 Stockwell transform

The S-transform introduced by Stockwell (1996), can be described as a special case of the wavelet transform, using a Morlet-type wavelet with phase and amplitude adjustments (Gibson 2006, Ventosa et al 2008). It uses a Gaussian window to balance temporal and spectral resolution. More importantly, frequency bands are phase-normalized, which means more precision at time scale, therefore, the input signal can be recovered from the transform in a lossless way if we trivialize numerical errors. The S-transform provides a time-frequency representation input for the ML model.

2. Methodology

2.1 The programming environment

Open sourced code was deliberately selected to facilitate accessibility to the following workflow, the core language is Python and it has been implemented on the Jupyter Notebook (IPython) Development Environment. Several Python modules are required in order to reproduce this work.

2.1.1 About Python and the necessary modules

Python is an interpreted high-level programming language for general-purpose programming. It was chosen for this work for more reasons than its accessibility as an open sourced language. Remarkably, nowadays 2018, Python stands among the most used languages for data science and scientific research. In Kaggle competitions (a renown online platform for data science in predictive modelling) Python is basically regarded as standard for building deep learning models. Furthermore Stack Overflow Developer Survey results from 2018 stated that “Python has a solid claim for being the fastest-growing major programming language”. Ergo, Python is proposed for the best fulfillment of these work objectives.

For the methodology of this work several Python modules have to be imported. Namely: MNE, NumPy, MatplotLib, TensorFlow and Keras.

NumPy is the fundamental package for scientific computing in Python.

Matplotlib is a Python 2D plotting library which produces publication quality figures.

TensorFlow is an open source software library for high performance numerical computation. Its flexible architecture allows easy deployment of computation across a variety of platforms (CPUs, GPUs and TPUs).

Keras is a high-level neural networks API, capable of running on top of TensorFlow.

MNE is a module for exploring, visualizing, and analyzing neuroscientific data, we take advantage of the time-frequency representations (TFR) for feature engineering.

The code generated in the following steps can be access at github.com/ralfcam/ML_MT

2.2 Synthetic data generation

Generating Electromagnetic time series can be done in python similarly to other scientific programming languages, for this step the NumPy module is essential. Working with synthetic data with a known spectro-temporal structure is crucial since it provides validation for the models. This procedure is described in the following steps, here will be presented the equations used in Python and the most relevant functions applied.

2.2.1 Declaring the physical parameters

Electromagnetic time series generation depend on the constant show in Table 1

Table 1. Parameters used in the signal generation

Parameter	Value	Unit
sampling frequency	1024	Hz
number of samples	1024	
magnetic declination	45	degrees
magnetic amplitude	10	nT
phase	0	degrees
resistivity of the space	100	ohm m
frequency of the signal	120	Hz

2.2.2 Generating the magnetic field (B)

The total magnetic field (Bt) is calculated along the polarization direction as a complex sinusoid using the Eq.3

$$B_t(t) = Be^{i(2\pi ft+\phi)} \quad \text{Eq. 3.}$$

where B is the magnitude of the magnetic field, f is the frequency, t the time and ϕ the phase. From Eq. 3 Bx and By components are the NS (x) and EW (y) components (Eq. 4)

$$B_x = B_t \cos\alpha\,;\; B_y = B_t \sin\alpha \quad \text{Eq. 4}$$

where α is the angle of the magnetic declination.

Next, Bx and By are transformed to the frequency domain with a fast Fourier transform FFT, applied with the numpy function “fft” as in the example:

```
Bx_f=np.fft.fft(Bx)
By_f=np.fft.fft(By)
```

2.2.3 Generating the impedance matrix (Z)

Given that the Resistivity of the homogeneous half-space is known, the computation of the impedance matrix can be done with forward calculation as Eq. 5

$$Z = \sqrt{5f\rho} * e^{i\pi/4} \quad \text{Eq. 5}$$

Where ρ is the resistivity of the homogeneous media, and f is a discretization of the frequency. Finally the elements of the impedance tensor are Eq.6

$$Zxy = Zs;\quad Zxx = 0 * Zs;\quad Zyy = Zxx;\quad Zyx = -Zs \quad \text{Eq. 6}$$

2.2.4 Generating the electric field (E)

To obtain Ex and Ey in the frequency domain the Eq. 7 is used.

$$\begin{pmatrix} Ex \\ Ey \end{pmatrix} = \begin{pmatrix} Zxx & Zxy \\ Zyx & Zyy \end{pmatrix} \begin{pmatrix} Bx \\ By \end{pmatrix} \quad \text{Eq. 7}$$

Then by applying an inverse FFT is possible to obtain the time representations. Here, an example of the code:

```
# Computation of Ex & Ey in frequency domain
Ex_f = Zxx*Bx_f + Zxy*By_f
Ey_f = Zyx*Bx_f + Zyy*By_f

# Computation of the time series for Ex & Ey
Ex = np.fft.ifft(Ex_f)
Ey = np.fft.ifft(Ey_f)
```

Figure 5 shows the synthetic data (electromagnetic components Bx, By, Ex, Ey)

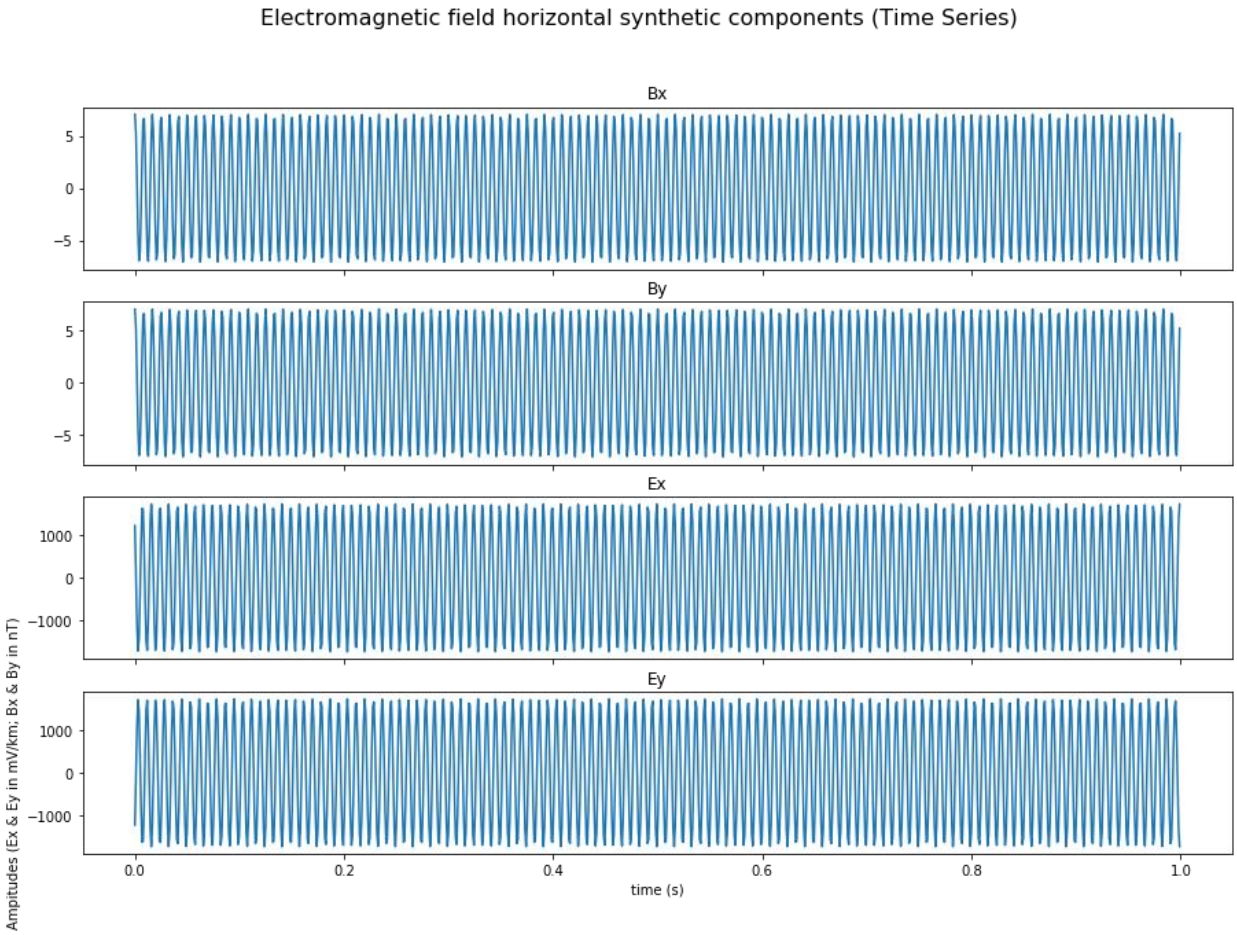

Figure 5 Electromagnetic field horizontal synthetic components (Time Series)

2.2.5 Generating the random noise and final data

Noise with a Gaussian distribution is added to the electromagnetic components to generate the final synthetic input data. The magnitude of the noise is escalated to a coefficient that depends on the amplitude of the signal; therefore it is possible to know the relation of signal amplitude versus noise amplitude for later assessment. For the first simulation, a coefficient of 0.3 was implemented; the data is shown in Figure 6.

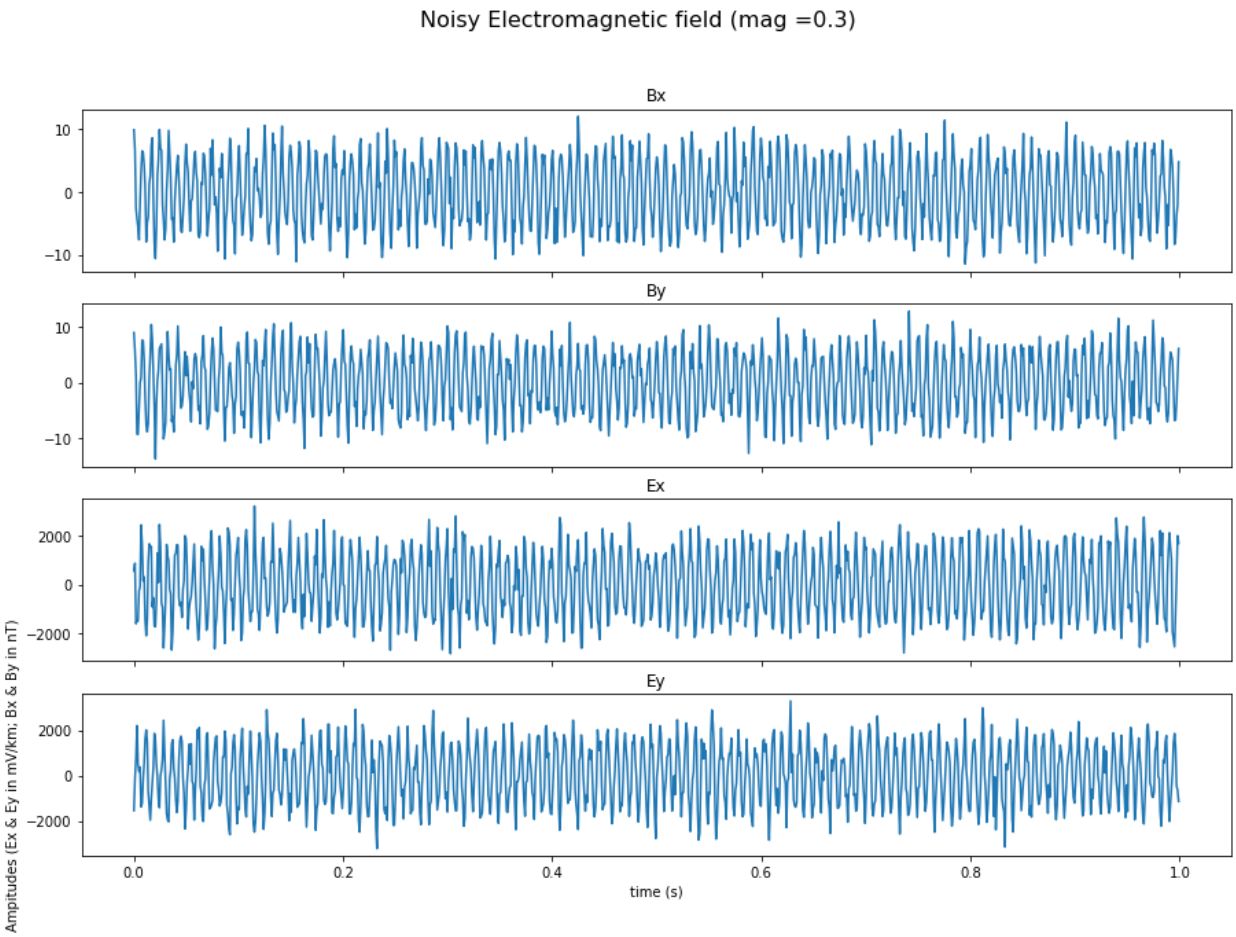

Figure 6 Noisy Electromagnetic field components

2.3 Feature Engineering

2.3.1 Transforming the data to time-frequency representations (TFR)

The MNE Python module has an already optimized function (“tfr_array_stockwell”) to return the power of the Stockwell transformed data; therefore it can be applied on the noisy

signal to obtain the TFR (Figure 7) for inputting into the machine learning model. In this case, the spectral / temporal resolution is controlled by specifying different widths of the Gaussian window using the width parameter. A width of 1 (classical S-Transform) was declared for the first simulation. The amplitude values (power) of the S-transform are then normalized for better performance of the machine learning model. The following is an example of the code.

```
### Calculate the Strasform / MNE array function
[st_power_Bx] = tfr_array_stockwell(data_Bx, sfreq, fmin, fmax)
```

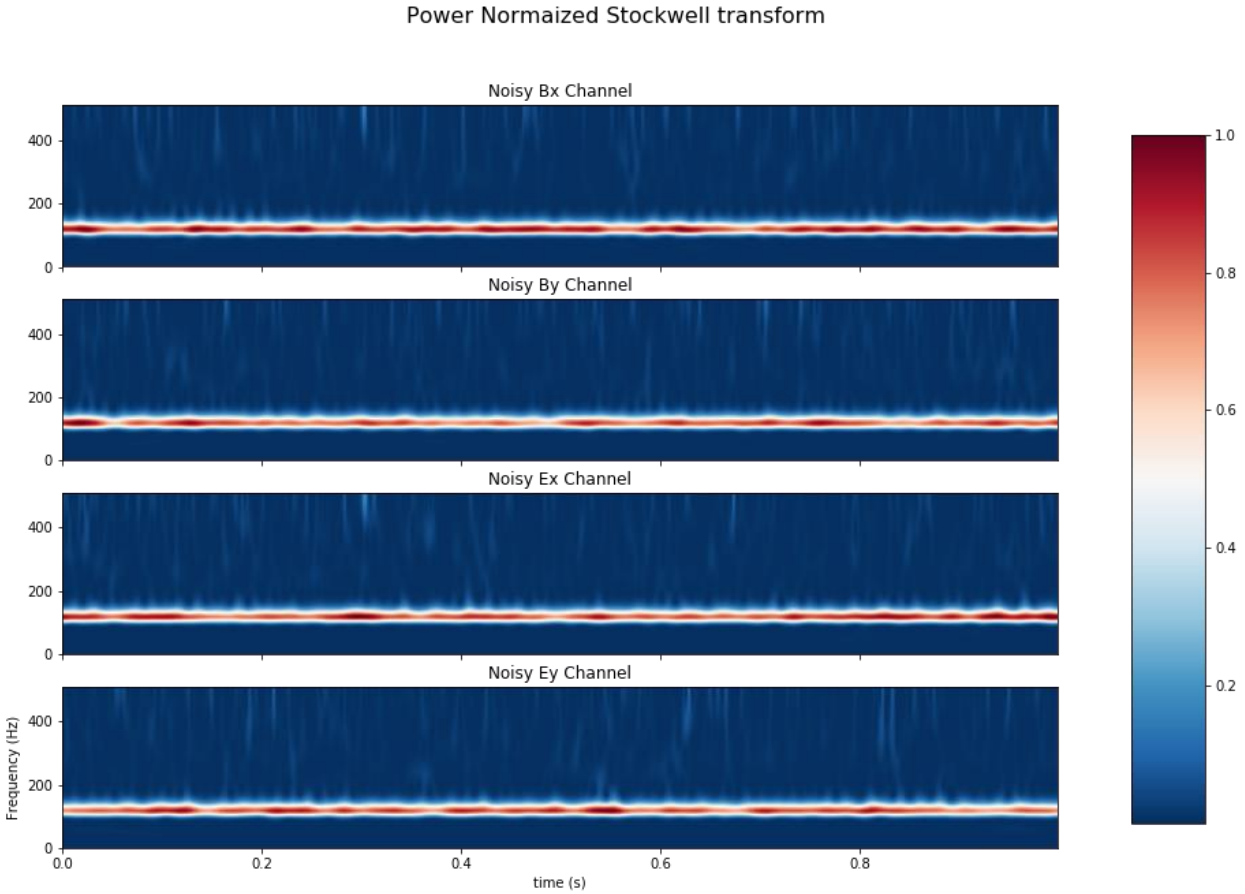

Figure 7 Power Normalized Stockwell Transform

2.4 Implementing the Machine Learning model

LSTM networks are predefined in Keras; the regression is implemented accordingly to the typical Keras workflow (Chollet 2017):

1. Define the training data: the normalized power matrices obtained from the S-Transform of the magnetic components (time series) Bx and By are set as input data (x_train, x_validation) and similarly the electric components Ex and Ey are set as input targets (y_train and y_validation).

2. Define a network of layers for the Machine Learning Model that maps your inputs to your targets, for the regression sequential layers are stack, each layer do the following:

    a. Apply the LSTM algorithm to the specified number of input dimensions (3D tensor with the shape defined by batch size, time steps and input dimension) Several LSTM can be stacked for achieving better results. In this case 3 LSTM layers were stacked.

    b. Densification of the LSTM network to a one neuron output with a linear activation function of dimensions equal to the number of row in the S-transform. Note that RNN allows for output dimensions to be different that inputs' therefore experimentation to improve time resolution can be done. The following is a simplified example of the implementation of the code

```
layers = [LSTM(…),
          LSTM(…),
          LSTM(…),
          Dense(…)]

model = Sequential(layers)
```

3. Configure the learning process by defining a mean squared error loss function, an 'rmsprop' optimizer, and the accuracy metric to monitor.

```
model.compile(loss='mean_squared_error',
              optimizer='rmsprop',
              metrics=['accuracy'])
```

4. Iterate on your training data by calling the fit() method. This takes as basic parameters: The training data, the batch size (set to 120) and the number of epochs (set to 30).

Then the prediction or validation of the Ex can be done with the function predict.

```
prediction_Ex = model.predict(x_val)
```

2.5 Methodology for rea data

To denoise real time series data a variation of the previous workflow was done, given that the idealization of a 1D homogenous half-space does not apply. To overcome the limitation of validation data a self-supervised learning model can be used (see Chollet F., 2017).

This variation consists on using the TFR of a channel for training data as well as for validation data, in other words, the model learns to predict future values of the time-series itself. This can be achieve by setting as input the time series and a shifted version of it, set in this case to 32 to equal the batch size of the LSTM model time step.

3. Results

Several simulations were run in order to compare the behavior of the machine learning model at different orders of magnitude of induced noise; here are presented the ones where hyperparameters adjusted to the most interesting results:

3.1 First simulation

The first simulation corresponds to the example case described in the methodology, where a coefficient of 0.3 maximum signal amplitude versus maximum non-coherent noise amplitude was used (at 120 Hz fundamental frequency); and the hyperparameter adjustment of a batch size of 32, a “statefull” parameter set to “True” (overlapping the input columns) and 30 epochs. The following results are obtained after approximately 5 minutes training on a regular CPU.

3.1.1 First simulation: Validation curves (MSE & Accuracy)

Figure 8 provides quantitative validation of the model, the mean squared error calculated between the magnetic and electric components (training and validation) decreases from 0.007 to less than 0.001, coherently the mapping accuracy tends to increase. Mapping accuracy starts

at around 0.2 and ends at 0.7 after 30 epochs (30 iterations over the 1024 time steps at 32 batches per step with overlapping).

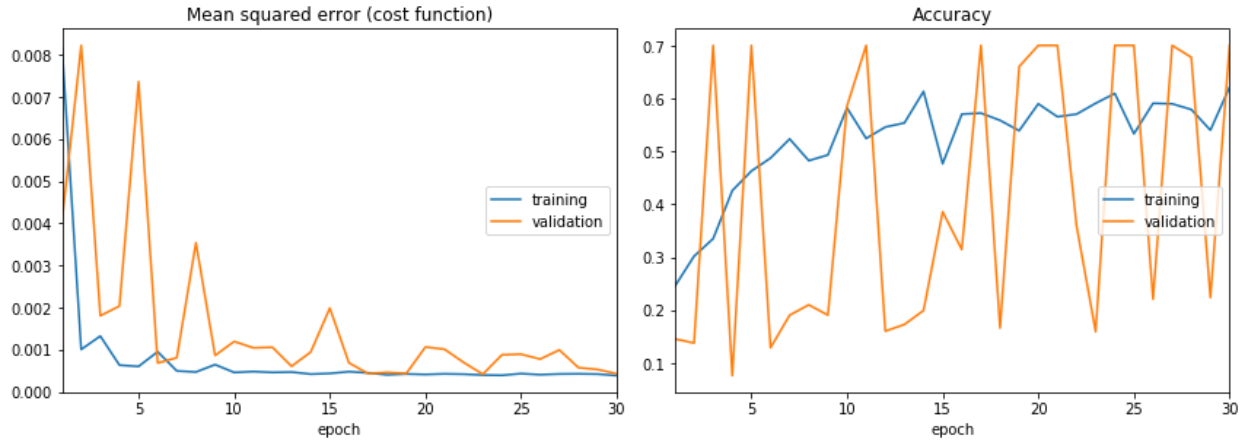

Figure 8 First simulation: Validation curves

3.1.2 First simulation: Model learned weights

Figure 9 is a plot of the weights learned by the last sequential layer (densification) after the 30 epochs. Weights values increase at the 120 Hz frequency, as expected from the synthetic data, generated at that same frequency before the addition of noise.

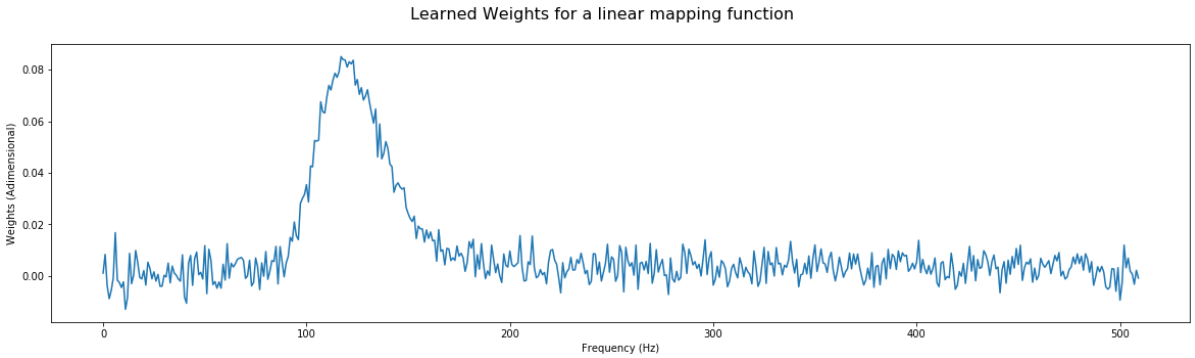

Figure 9 First simulation: Model learned weights

3.1.3 First simulation: Predict denoised electric component

Figure 10 First simulation: Predict denoised electric component depicts the normalized S-transform of the generated electric component (Ex) with an added noise of magnitude 0.3 times higher than the signal's. It also shows the transformations of the predicted/denoised same channel by the ML model. It can be seen that he 120 Hz is enhanced meanwhile the added noise (other-than 120 Hz frequencies) is reduced.

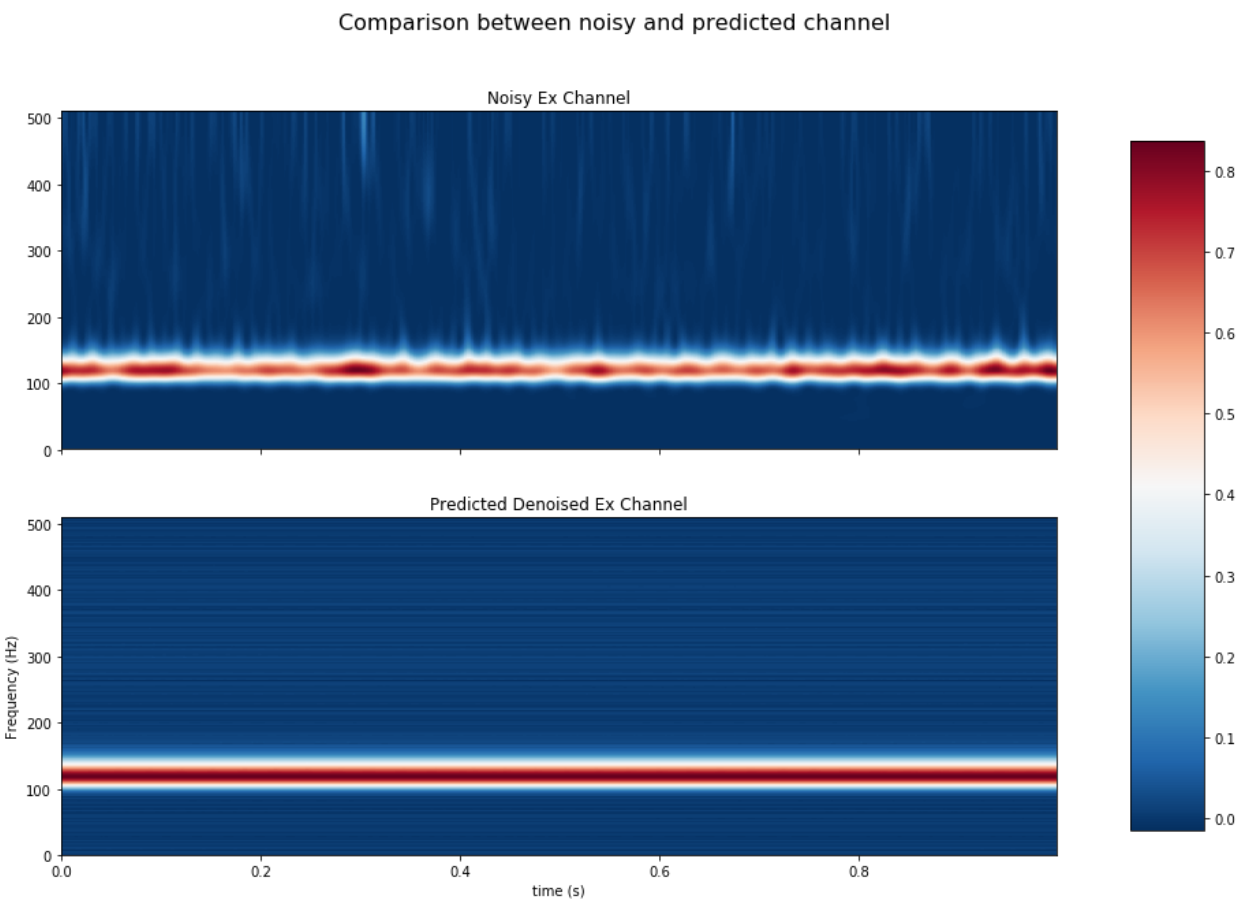

Figure 10 First simulation: Predict denoised electric component

3.2 Second simulation

Resetting the noise coefficient to 1, and keeping the hyperparameters unchanged:

3.2.1 Second simulation: Validation curves (MSE & Accuracy)

In Figure 11 the MSE decreases from 0.007 to almost the half of that, but only in the training curve (blue), the validation curve moves chaotically. The accuracy increases, but then again only for the training curve, starting at around 0.16 and ends at 0.22 after 30 epochs.

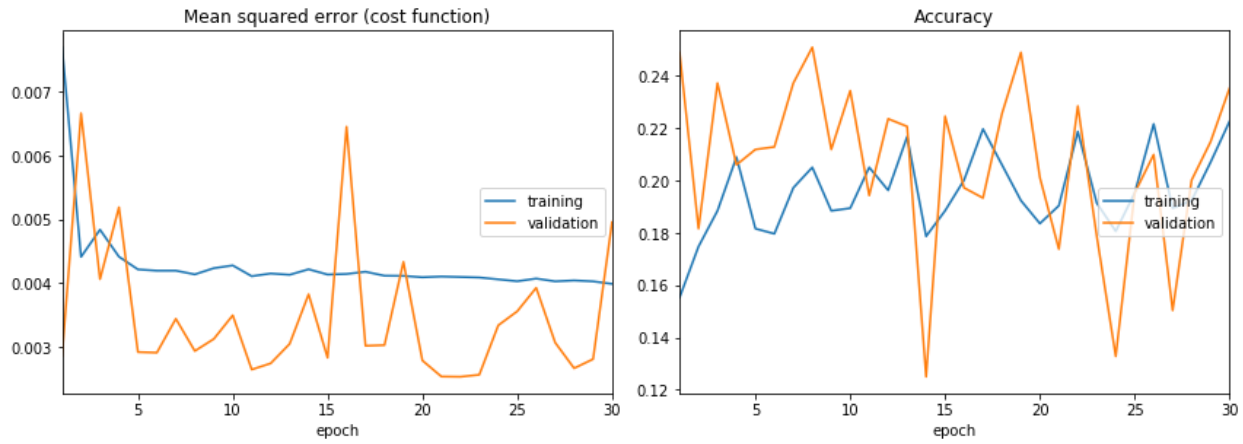

Figure 11 Second simulation: Validation curves (MSE & Accuracy)

3.2.2 Second simulation: Model learned weights

In Figure 11 the weights show again the peek at the 120 Hz frequency as expected, but a slight trend up to the higher frequencies appears as the noise fails to be completely removed.

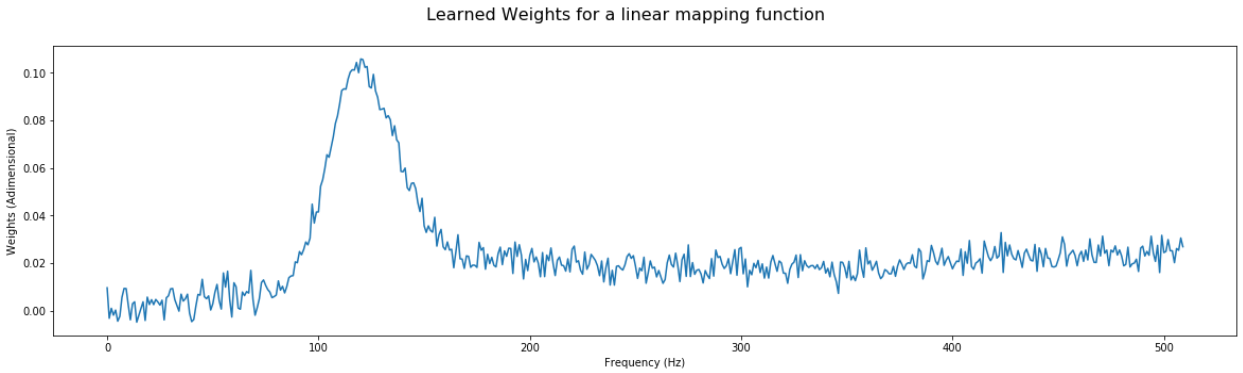

Figure 12 Second simulation: Model learned weights

3.2.3 Second simulation: Predict denoised electric component

Figure 11 normalized S-transform of the generated electric component (Ex) with an added noise of magnitude 1 time higher than the signal's. It also shows the transformations of the predicted/denoised same channel by the ML model. It can be seen that the 120 Hz is still evident although the noise in this occasion is not fully removed.

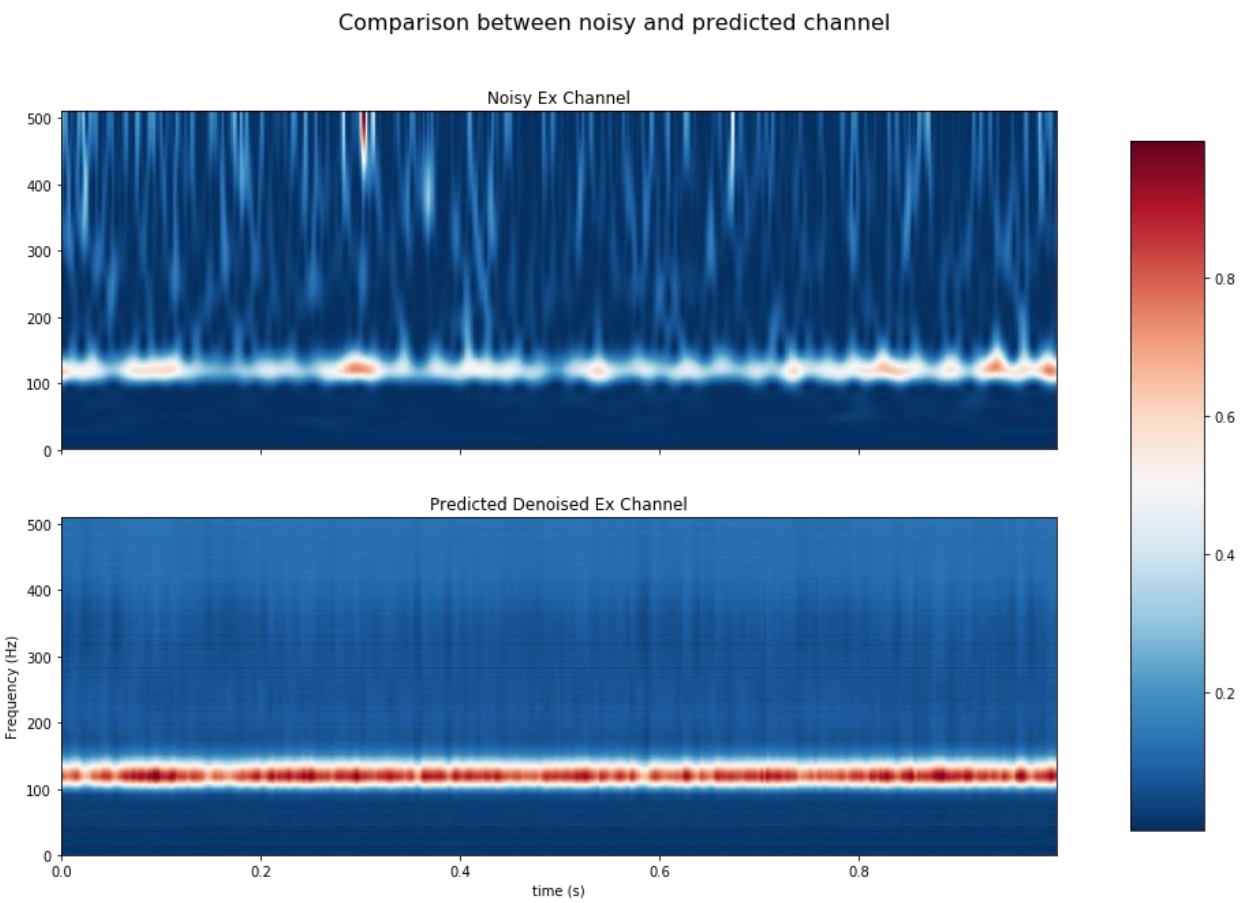

Figure 13 Second simulation: Predict denoised electric component

3.3 Real data denoised prediction

The real data denoise was done over a subset (1024 of 225279 samples) of a MT station sampled at 4096 Hz. For this a hyperparameter adjustment was to increase the epochs to 100, also the learning showed better results without the normalization of the S-transform and the Gaussian width set to 3. The results are obtained after ~15 minutes training.

3.3.1 Predict denoised electric component

The model prediction leads to an expression of the signal with a fundamental frequency that peaks at ~1600 Hz as can be seen in Figure 14.

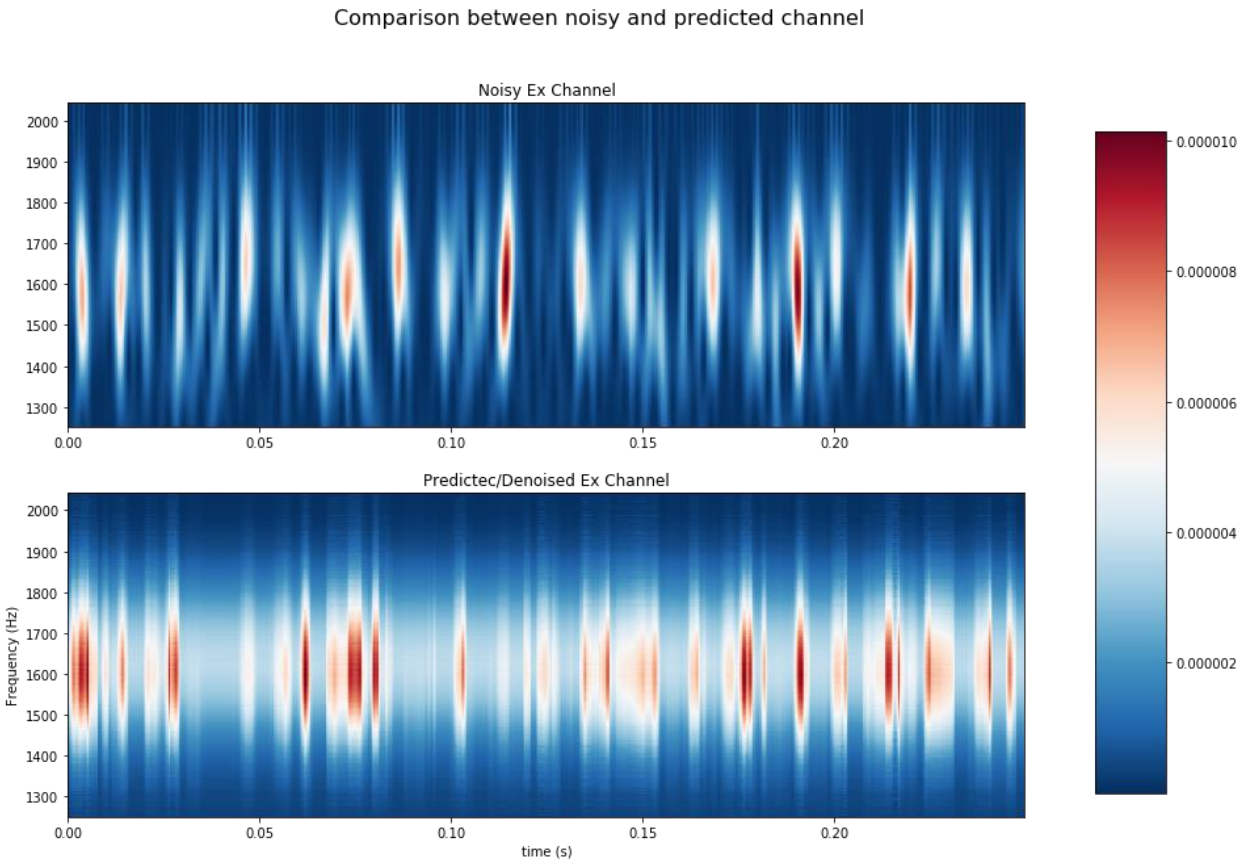

Figure 14 Predict denoised electric component of real data

4. Discussion

Results suggest that the ML approach to denoise every component is accurate; it is possible to denoise every channel by systematical assigning the corresponding combination of training data to each channel. The weights obtained from the ML regression model could be further used to stablish a workflow that leads to retrieving the apparent resistivities of the strata.

As it can be inferred from the values of the validation graphs and the S-transforms plots of the denoised channel, the methodology is compromised as the noise to signal proportion increases, as it could be expected. Although readjusting the hyperparameters or/and providing more input data seems to compensate until a certain point, when the noise magnitude gets to be many times higher than the signal's. In this work synthetic EM data was effectively denoised to a proportion of 3 times the magnitude of the noise to the signal´s. For higher magnitudes of noise other approaches can then be tested, such as improving the feature engineering, for example, finding the most appropriate length of the Gaussian window of the applied S-transform or using a non-linear frequency scale, also, evaluating different TFR. Although, as expected, the

S-transform provides an effective approach; it success in depicting the declared fundamental frequency of the synthetic generated data.

It has to be kept I in mind that the models works on the assumption of the empirical physical relations of the electromagnetic field, finding a mapping function between the magnetic and electrical components. Further experimentation can be done to implement a variation of the proposed model for a 2D inversion case, using the input tensor containing the real and the imaginary components of the TFR, therefore retrieving a complex mapping function that can be related more directly to the impedance tensor.

The mapping function finds correlation for coherent noise among the channels; this influence could be further studied. Proposedly, alternative methods may be more efficient to reduce to a better margin the signal to noise ratio; architected with deeper neural network or/and more pioneering models such as convolutional LSTM networks, Bidirectional RNN and so on. In particular, to reduce coherent noise reinforcement learning and self-supervised models could provide a valuable approach.

The use of Python as proposed for the task proved to be advantageous, most of the required tools are already to be found very well optimized in native libraries. It is a huge benefit to have libraries such as TensorFlow that can be used for optimal numerical processing on a CPU (CUDA / CuDNN) and better yet a GPU (BLAS / Eigen), this should lead geo-data-science in parallel with up to date advances in machine learning. The extensibility of the notions that ML can bring to geosciences research are noted.

5. Conclusions

A successful methodology was established to improve the signal to noise ration of the Electromagnetic component with Machine Learning, specifically with Long Short-Term Memory Networks (LSTM). The methodology tested in real data provides a clearer representation of the signal.

In this methodology is important to use a time frequency representation as an input tensor for the Machine Learning model. This enables the use of the physical relations between the electromagnetic fields which is in the frequency domain B=Z*E. The implementation of the S-

Transform is convenient because it is an open source code in the python module MNE and it has been proved that provides a quality time frequency representation of the generated synthetic data.

The application of the LSTM successfully achieved an effective representation of the fundamental frequency declared for the synthetic generated signal (120 Hz) taking as an input the signal with the induced Gaussian noise (up to three times the magnitude of the noise vs the magnitude of the signal). Therefore the LSTM Network model has been proved a feasible method to enhance the signal to noise ratio.

Modified versions of the proposed workflow can be further applied in similar time-series analysis problems. Mainly, in controlled sourced electromagnetic methods which depend upon similar physical relations as the one studied in this work.

The proposed methodology was implemented on a regular Central Processing Unit (CPU).It could be many times more computing cost-effective if it would be implemented on a Graphic Processing Unit (GPU) or even better in a Tensor Processing Units (TPU).

This work validates the capacity of Machine Learning to improve the achievements of data analysis in geoscience.

6. Acknowledgement

This work was highly influenced by the mentorship and ideas of Prof. Alex Marcuello. Who always received and encouraged my motivation, even though I was always late for his appointments. Thanks for that. He also provided segments of the code used in this work and brought valuable insight.

7. References

Banks, R. J. (1998). The effects of non-stationary noise on electromagnetic response estimates. Geophysical Journal International, 135(2), 553-563.

Bengio, Y., Frasconi, P., and Simard, P. (1993). *The problem of learning long-term dependencies in recurrent networks. In IEEE International Conference on Neural Networks, pages 1183–1195, San Francisco. IEEE Press. (invited paper). 403*

Bengio, Y., Simard, P., and Frasconi, P. (1994). *Learning long-term dependencies with gradient descent is difficult. IEEE Tr. Neural Nets . 18, 401, 403, 411*

Bielecka, M., Danek, T., Wojdyla, M., & Baran, G. (2009). Neural networks application to reduction of train caused distortions in magnetotelluric measurement data. Schedae Informaticae, 17(18), 75-86.

Bishop, C. M. (1995). *Training with noise is equivalent to Tikhonov regularization. Neural computation, 7(1), 108-116.*

Cagniard, L. (1953). Basic theory of the magneto-telluric method of geophysical prospecting. Geophysics, 18(3), 605-635.

Calderón-Macías, C., Sen, M. K., & Stoffa, P. L. (2000). *Artificial neural networks for parameter estimation in geophysics. Geophysical Prospecting, 48(1), 21-47.*

Carbonari, R., D'Auria, L., Di Maio, R., & Petrillo, Z. (2017). *Denoising of magnetotelluric signals by polarization analysis in the discrete wavelet domain. Computers & Geosciences, 100, 135-141.*

Chatfield, C. (2016). *The analysis of time series: an introduction. CRC press*

Chave, A. D., & Jones, A. G. (Eds.). (2012). The magnetotelluric method: Theory and practice. Cambridge University Press.

Chollet, F. (2017). Deep learning with python. Manning Publications Co.

Cottrell, M., Girard, B., Girard, Y., Mangeas, M., & Muller, C. (1995). *Neural modeling for time series: a statistical stepwise method for weight elimination. IEEE transactions on neural networks, 6(6), 1355-1364.*

Dehghani, M. (2011, December). *Applications of Electromagnetic Measurement Methods in Oil and Gas Industry. In AIP Conference Proceedings (Vol. 1414, No. 1, pp. 95-101). AIP.*

Escalas, M., Queralt, P., Ledo, J., & Marcuello, A. (2013). *Polarisation analysis of magnetotelluric time series using a wavelet-based scheme: A method for detection and characterisation of cultural noise sources. Physics of the Earth and Planetary Interiors, 218, 31-50.*

Fontes, S. L., Harinarayana, T., Dawes, G. J. K., & Hutton, V. R. S. (1988). *Processing of noisy magnetotelluric data using digital filters and additional data selection criteria. Physics of the earth and planetary interiors, 52(1-2), 30-40.*

Gibson, P. C., Lamoureux, M. P., & Margrave, G. F. (2006). *Letter to the editor: Stockwell and wavelet transforms. Journal of Fourier Analysis and Applications, 12(6), 713-721.*

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning (adaptive computation and machine learning series). Adaptive Computation and Machine Learning series, 800.*

Graves, A. (2012). *Supervised sequence labelling. In Supervised sequence labelling with recurrent neural networks (pp. 5-13). Springer, Berlin, Heidelberg.*

Graves, A., Mohamed, A. R., & Hinton, G. (2013). *Speech recognition with deep recurrent neural networks. In proceedings of the 2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP),(pp. 6645-6649).*

Hochreiter, S. (1991). *Untersuchungen zu dynamischen neuronalen Netzen. Diploma, Technische Universität München, 91, 1.*

Hochreiter, S., & Schmidhuber, J. (1997). *Long short-term memory. Neural computation, 9(8), 1735-1780.*

Junge, A. (1996). *Characterization of and correction for cultural noise. Surveys in Geophysics, 17(4), 361-391.*

Manoj, C., & Nagarajan, N. (2003). *The application of artificial neural networks to magnetotelluric time-series analysis. Geophysical Journal International, 153(2), 409-423.*

Oettinger, G., Haak, V., & Larsen, J. C. (2001). *Noise reduction in magnetotelluric time-series with a new signal–noise separation method and its application to a field experiment in the Saxonian Granulite Massif. Geophysical Journal International, 146(3), 659-669.*

Parker, R. L. (1983). *The magnetotelluric inverse problem. Geophysical surveys, 6(1-2), 5-25.*

Popova, I. V., & Ogawa, Y. (2007). *Application of a modified Hopfield neural network to noisy magnetotelluric data. Izvestiya, Physics of the Solid Earth, 43(3), 217-224.*

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986)*. Learning representations by back-propagating errors. nature, 323(6088), 533.*

Stockwell, R. G., Mansinha, L., & Lowe, R. P. (1996). *Localization of the complex spectrum: the S transform. IEEE transactions on signal processing, 44(4), 998-1001.*

Van der Baan, M., & Jutten, C. (2000). *Neural networks in geophysical applications. Geophysics, 65(4), 1032-1047.*

Ventosa, S., Simon, C., Schimmel, M., Dañobeitia, J. J., & Mànuel, A. (2008). *The $ S $-transform from a wavelet point of view. IEEE Transactions on Signal Processing, 56(7), 2771-2780.*

Vozoff, K. (1991). *The magnetotelluric method. In Electromagnetic Methods in Applied Geophysics: Volume 2, Application, Parts A and B (pp. 641-712). Society of Exploration Geophysicists.*

Weckmann, U., Magunia, A., & Ritter, O. (2005). *Effective noise separation for magnetotelluric single site data processing using a frequency domain selection scheme. Geophysical Journal International, 161(3), 635-652*

Werbos, P. J. (1990). *Backpropagation through time: what it does and how to do it. Proceedings of the IEEE, 78(10), 1550-1560.*

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., ... & Klingner, J. (2016). *Google's neural machine translation system: Bridging the gap between human and machine translation. arXiv preprint arXiv:1609.08144.*