

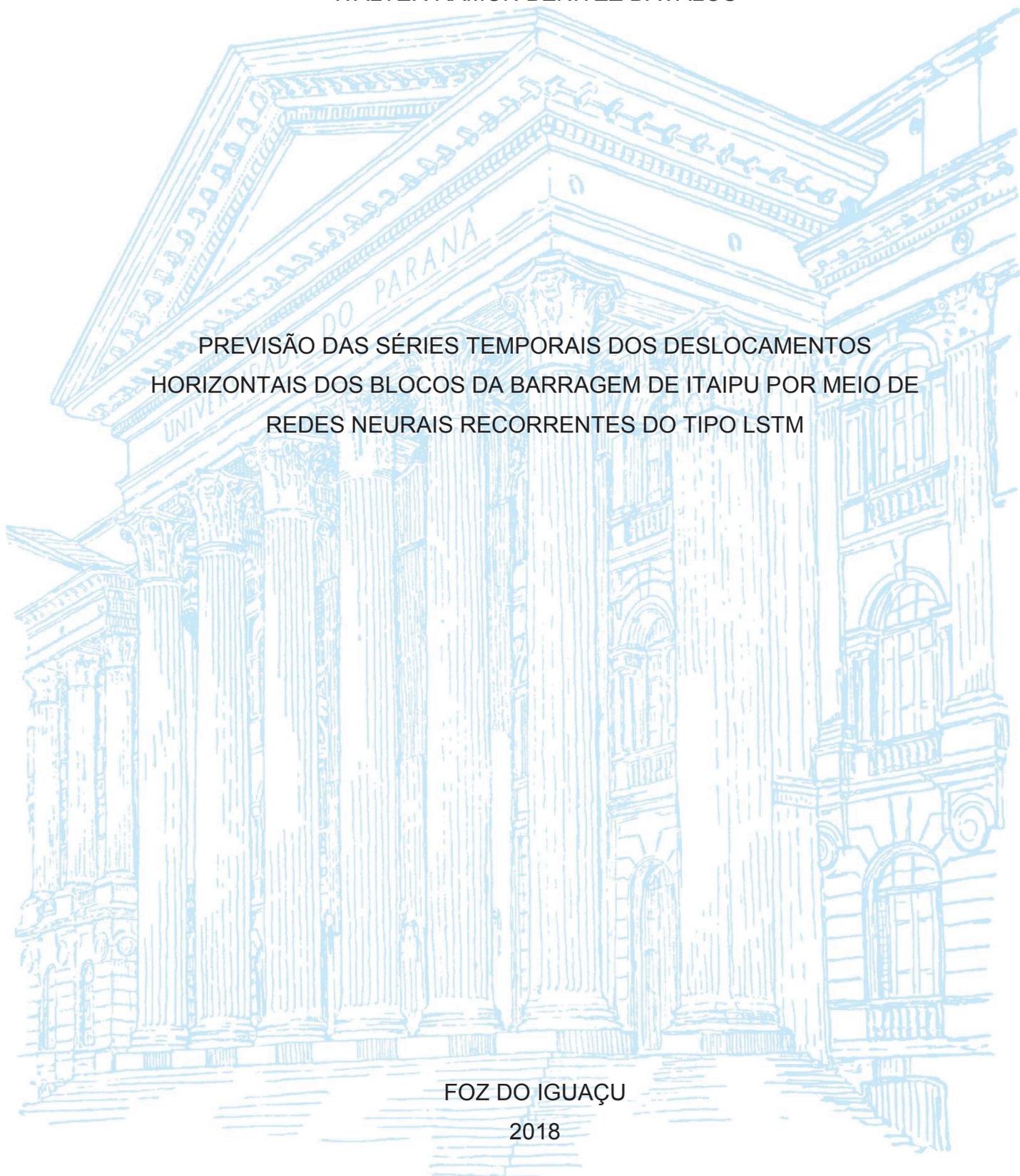
UNIVERSIDADE FEDERAL DO PARANÁ

WALTER RAMON BENÍTEZ DÁVALOS

PREVISÃO DAS SÉRIES TEMPORAIS DOS DESLOCAMENTOS  
HORIZONTAIS DOS BLOCOS DA BARRAGEM DE ITAIPU POR MEIO DE  
REDES NEURAS RECORRENTES DO TIPO LSTM

FOZ DO IGUAÇU

2018



WALTER RAMON BENÍTEZ DÁVALOS

PREVISÃO DAS SÉRIES TEMPORAIS DOS DESLOCAMENTOS  
HORIZONTAIS DOS BLOCOS DA BARRAGEM DE ITAIPU POR MEIO DE  
REDES NEURAS RECORRENTES DO TIPO LSTM

Monografia apresentada ao curso de Pós-Graduação em Métodos Numéricos em Engenharia, Setor de Tecnologia e o Setor de Ciências Exatas, Universidade Federal do Paraná, como requisito parcial à obtenção do título de Especialista em Métodos Numéricos em Engenharia.

Orientador: Prof. Dr. Jairo Marlon Corrêa

Coorientadores: Prof. Dr. Anselmo Chaves Neto  
Prof. Dr. Samuel Bellido Rodrigues

FOZ DO IGUAÇU

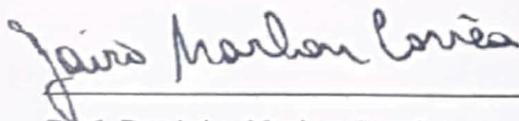
2018

## TERMO DE APROVAÇÃO

WALTER RAMON BENÍTEZ DÁVALOS

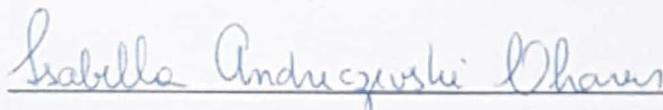
### PREVISÃO DAS SÉRIES TEMPORAIS DOS DESLOCAMENTOS HORIZONTAIS DOS BLOCOS DA BARRAGEM DE ITAIPU POR MEIO DE REDES NEURAIS RECORRENTES DO TIPO LSTM

Monografia aprovada como requisito parcial à obtenção do título de Especialista em Ciências com Área de Concentração em Programação Matemática, Curso de Especialização em Métodos Numéricos em Engenharia, Setor de Ciências Exatas, Universidade Federal do Paraná, pela seguinte banca examinadora:



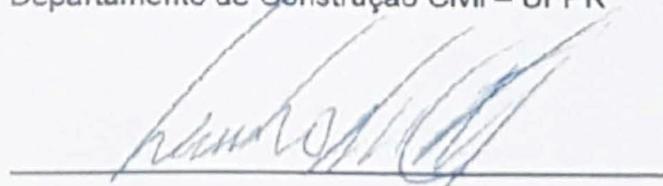
Prof. Dr. Jairo Marlon Corrêa

Orientador – Departamento de Matemática – UTFPR



Prof.<sup>a</sup> Dr.<sup>a</sup> Isabella Andreczewski Chaves

Departamento de Construção Civil – UFPR



Prof. Dr. Levi Lopes Teixeira

Departamento de Matemática – UTFPR

Foz do Iguaçu, 10 de dezembro de 2018.

Dedicado especialmente à minha família, amigos e aos meus orientadores que me apoiaram em todo momento na realização deste trabalho.

## **AGRADECIMENTOS**

Agradecimento público ao PTI, ao CEASB e a UFPR pela oportunidade dada para cursar esta especialização e a os meus companheiros da especialização pelo apoio obtido ao longo do curso.

Agradeço também ao PTI-PY que me concedeu o tempo e apoio para realizar esta especialização e ao meu grupo de trabalho no CTIC que em todo momento me apoiaram em tudo.

E a minha família que é o pilar que me mantem em pé.

Muito obrigado a todos vocês, sem o apoio de vocês isto não seria possível.

“A melhor característica de um profeta é ter uma boa memória” (George Savile, 1º Marques de Halifax)

## RESUMO

O monitoramento da saúde estrutural da barragem é um dos assuntos de maior importância no âmbito da segurança das mesmas. Dados como temperatura, deslocamento, nível da água são vitais para o correto planejamento dos trabalhos de manutenção. Neste trabalho foram obtidas as previsões dos dados de deslocamento horizontal dos blocos F05, F13, e F19 nos eixos X (na direção do fluxo do rio) e Y (perpendicular ao fluxo do rio), obtidos pelos pêndulos diretos instalados na estrutura da Usina Hidrelétrica da Itaipu Binacional por meio do uso de redes neurais artificiais recorrentes do tipo *Long Short Term Memory (LSTM)*. Este tipo de redes permite melhor utilização dos dados passados para a previsão, devido a seu caráter recorrente e o uso de células de memória. Com isto, um modelo foi proposto com uma busca por grade para refinar melhor os hiperparâmetros. Os resultados obtidos demonstram a eficiência da técnica quando utilizada nas referidas séries temporais quando comparada com técnicas como ARIMA e Redes Neurais MLP. Finalmente, pode-se concluir que as redes neurais recorrentes são promissoras para a área de previsões das séries temporais e o modelo proposto deve ser testado em mais dados de séries temporais.

Palavras-chave: RNA. LSTM. RNR. Segurança. Previsão. Modelagem.

## **ABSTRACT**

The monitoring of the structural health of the dam is one of the most critical issues in dam safety. Data such as temperature, displacements, water level, among others are vital to the correct planning of maintenance. In this work, data from direct pendulums installed inside Itaipu Hydroelectric Power Plant is used to forecast horizontal displacement of F05, F13 and F19 blocks in the X-axes (in the direction of river flow) and Y-axes (perpendicular to river flow) with recurrent neural networks of the type Long Short Term Memory (LSTM). This type of network allows better use of the data due to its recurrent character and the use of memory cells. A model architecture was proposed with a grid search to tune its hyperparameters. The forecast results demonstrate the efficiency of the technique for these particular time series when compared to ARIMA and MLP. Finally, we conclude that using recurrent neural networks for time series is promising, and the proposed model should be tested with more time series data.

Keywords: ANN. LSTM. RNN. Security. Forecasting. Modeling.

## LISTA DE FIGURAS

FIGURA 1 INSTRUMENTAÇÃO DA BARRAGEM.....	17
FIGURA 2 ESQUEMA SIMPLIFICADO DE UM NEURÔNIO .....	21
FIGURA 3 PERCEPTRON SIMPLES DE UMA CAMADA .....	22
FIGURA 4 PERCEPTRON DE MÚLTIPLAS CAMADAS (MLP).....	23
FIGURA 5 EXEMPLO DE TREINAMENTO SUPERVISIONADO .....	25
FIGURA 6 EXEMPLO DE TREINAMENTO NÃO SUPERVISIONADO.....	26
FIGURA 7 REDE NEURAL RECORRENTE BÁSICA .....	27
FIGURA 8 RNR DE MÚLTIPLAS CÉLULAS E MÚLTIPLOS NEURÔNIOS.....	27
FIGURA 9 ARQUITETURAS BÁSICAS DAS REDES NEURAIIS RECORRENTES .	29
FIGURA 10 ESTRUTURA DE UMA UNIDADE LSTM .....	30
FIGURA 11 ESTRUTURA PROPOSTA .....	34
FIGURA 12 FLUXOGRAMA DO TREINAMENTO E ESCOLHA DOS MODELOS NEURAIIS .....	37
FIGURA 13 REDE NEURAL DE DUAS CAMADAS.....	67

## LISTA DE GRAFICOS

GRÁFICO 1 SÉRIE TEMPORAL F05X.....	39
GRÁFICO 2 FAC E FACP DA SÉRIE F05X.....	39
GRÁFICO 3 SÉRIE TEMPORAL F05Y .....	41
GRÁFICO 4 FAC E FACP DA SÉRIE F05Y.....	41
GRÁFICO 5 SÉRIE TEMPORAL F13X.....	42
GRÁFICO 6 FAC E FACP DA SÉRIE F13X.....	43
GRÁFICO 7 SÉRIE TEMPORAL F13Y .....	44
GRÁFICO 8 FAC E FACP DA SÉRIE F13Y.....	44
GRÁFICO 9 SÉRIE TEMPORAL F19X.....	45
GRÁFICO 10 FAC E FACP DA SÉRIE F19X.....	46
GRÁFICO 11 SÉRIE TEMPORAL F19Y .....	47
GRÁFICO 12 FAC E FACP DA SÉRIE F19Y.....	48
GRÁFICO 13 PREVISÕES DA SÉRIE F05X .....	50
GRÁFICO 14 PROGRESSÃO DO ERRO PORCENTUAL ABSOLUTO NA F05X....	50
GRÁFICO 15 FAC DOS RESÍDUOS DAS PREVISÕES DA SÉRIE F05X .....	51
GRÁFICO 16 DAS PREVISÕES DA SÉRIE F05Y.....	52
GRÁFICO 17 PROGRESSÃO DO ERRO PORCENTUAL ABSOLUTO NA F05Y....	52
GRÁFICO 18 FAC DOS RESÍDUOS DAS PREVISÕES DA SÉRIE F05Y .....	53
GRÁFICO 19 PREVISÕES DA SÉRIE F13X.....	54
GRÁFICO 20 PROGRESSÃO DO ERRO PORCENTUAL ABSOLUTO NA F13X....	54
GRÁFICO 21 FAC DOS RESÍDUOS DAS PREVISÕES DA SÉRIE F13X .....	55
GRÁFICO 22 PREVISÕES DA SÉRIE TEMPORAL F13Y.....	56
GRÁFICO 23 PROGRESSÃO DO ERRO PORCENTUAL ABSOLUTO NA F13Y....	56
GRÁFICO 24 FAC DOS RESÍDUOS DAS PREVISÕES DA SÉRIE F13Y .....	57
GRÁFICO 25 PREVISÕES DA SÉRIE F19X.....	58
GRÁFICO 26 PROGRESSÃO DO ERRO PORCENTUAL ABSOLUTO NA F19X....	58
GRÁFICO 27 FAC DOS RESÍDUOS DAS PREVISÕES DA SÉRIE F19X .....	59
GRÁFICO 28 PREVISÕES DA SÉRIE F19Y .....	60
GRÁFICO 29 PROGRESSÃO DO ERRO PORCENTUAL ABSOLUTO NA F19Y....	60
GRÁFICO 30 FAC DOS RESÍDUOS DAS PREVISÕES DA SÉRIE F19Y .....	61

## LISTA DE QUADROS

QUADRO 1 SÉRIES DE DESLOCAMENTO DOS BLOCOS-CHAVE DA BARRAGEM PRINCIPAL DA USINA HIDRELÉTRICA DE ITAIPU. ...	32
QUADRO 2 HIPERPARÂMETROS DOS MODELOS .....	35

## LISTA DE TABELAS

TABELA 1 DADOS DESCRITIVOS DA SÉRIE TEMPORAL F05X.....	38
TABELA 2 DADOS DESCRITIVOS DA SÉRIE TEMPORAL F05Y.....	40
TABELA 3 DADOS DESCRITIVOS DA SÉRIE TEMPORAL F13X.....	42
TABELA 4 DADOS DESCRITIVOS DA SÉRIE TEMPORAL F13Y.....	43
TABELA 5 DADOS DESCRITIVOS DA SÉRIE TEMPORAL F19X.....	45
TABELA 6 DADOS DESCRITIVOS DA SÉRIE TEMPORAL F19Y.....	47
TABELA 7 RESULTADOS DO MODELADO DA SÉRIE F05X.....	49
TABELA 8 RESULTADOS DO MODELADO DA SÉRIE F05Y.....	51
TABELA 9 RESULTADOS DO MODELADO DA SÉRIE F13X.....	53
TABELA 10 RESULTADOS DO MODELADO DA SÉRIE F13Y.....	55
TABELA 11 RESULTADOS DO MODELADO DA SÉRIE F19X.....	57
TABELA 12 RESULTADOS DO MODELADO DA SÉRIE F19Y.....	59

## LISTA DE SIGLAS

RNR – Redes Neurais Recorrentes

RNA – Redes Neurais Artificiais

LSTM – *Long Short Term Memory*

MLP – *Multi Layer Perceptron*

SLP – *Single Layer Perceptron*

ARIMA – *Autoregressive Integrated Moving Average*

MAE – Erro absoluto médio

MAPE – Erro absoluto porcentual médio

MSE – Erro quadrático médio

FAC – Função de Autocorrelação

FACP – Função de Autocorrelação Parcial

ADAM – *Adaptive Moment Estimator*

GD – *Gradient Descent*

SGD – *Stochastic Gradient Descent*

GARCH – *Autoregressive conditional heteroskedasticity*

BPTT – *Backpropagation Through Time*

## LISTA DE SÍMBOLOS

$L$  – Número total de camadas da rede

$l$  – Indexação da camada da rede

$j$  – Indexação dos neurônios da camada  $l$  da rede

$k$  – Indexação dos neurônios da camada  $l - 1$  da rede

$o_j$  – Valor de referência do neurônio  $j$  na última camada  $L$  para um dado de treinamento

$Ct$  – Custo total da rede para um dado de treinamento

$W_{kj}^l$  – Peso sináptico que conecta o neurônio  $k$  da camada  $l - 1$  com o neurônio  $j$  na camada  $l$

$W^l$  – Matriz dos pesos sinápticos da camada  $l$

$Z_j^l$  – Entrada do neurônio  $j$  na camada  $l$

$Z^l$  – Vetor entrada dos neurônio na camada  $l$

$f_l$  – Função de ativação utilizada na camada  $l$

$a_j^l$  – Saída da função de ativação do neurônio  $j$  na camada  $l$

$a^l$  – Vetor das saídas da função de ativação na camada  $l$

$b_j^l$  – Desvio da combinação linear da entrada no neurônio  $j$  na camada  $l$

$b^l$  – Vetor de desvios da combinação linear da entrada na camada  $l$

$x_j$  – Entrada  $j$  da rede

$x_t$  – Entrada da rede dependente do tempo  $t$

$y_j$  – Saída da  $j$  rede

$\sigma$  – Função de ativação sigmoide

$\tanh$  – Função de ativação tangente hiperbólico

$in_t$  – Função da comporta de entrada da LSTM para a célula correspondente ao tempo  $t$

$out_t$  – Função da comporta de saída da LSTM para a célula correspondente ao tempo  $t$

$for_t$  – Função da comporta de esquecimento da LSTM para a célula correspondente ao tempo  $t$

$h_t$  – Vetor de saída da célula  $t$  da LSTM

$c_t$  – Vetor de estado da célula  $t$  da LSTM

$p$  – Último elemento de um vetor

$p_l$  – Último neurônio da camada  $l$

## SUMÁRIO

<b>1 INTRODUÇÃO</b> .....	<b>16</b>
1.1 OBJETIVOS .....	18
1.1.1 Objetivo geral .....	18
1.1.2 Objetivos específicos.....	18
1.2 JUSTIFICATIVA .....	18
<b>2 REVISÃO DE LITERATURA</b> .....	<b>19</b>
2.1 REDES NEURAIS ARTIFICIAIS (RNA).....	20
2.1.1 Características de uma RNA.....	24
2.1.2 Treinamento de Redes Neurais.....	25
2.1.3 Redes Neurais Recorrentes .....	26
2.1.4 O Problema da fuga de gradientes.....	29
2.1.5 Redes do tipo LSTM.....	30
<b>3 METODOLOGIA</b> .....	<b>32</b>
<b>4 ANÁLISE DAS SÉRIES TEMPORAIS</b> .....	<b>38</b>
4.1 SÉRIE TEMPORAL DO DESLOCAMENTO DO BLOCO F05 NO EIXO X .....	38
4.2 SÉRIE TEMPORAL DO DESLOCAMENTO DO BLOCO F05 NO EIXO Y .....	40
4.3 SÉRIE TEMPORAL DO DESLOCAMENTO DO BLOCO F13 NO EIXO X .....	42
4.4 SÉRIE TEMPORAL DO DESLOCAMENTO DO BLOCO F13 NO EIXO Y .....	43
4.5 SÉRIE TEMPORAL DO DESLOCAMENTO DO BLOCO F19 NO EIXO X .....	45
4.6 SÉRIE TEMPORAL DO DESLOCAMENTO DO BLOCO F19 NO EIXO Y .....	46
4.7 CONCLUSÃO DA ANÁLISE.....	48
<b>5 RESULTADOS</b> .....	<b>49</b>
5.1 MODELAGEM DA SÉRIE DO DESLOCAMENTO DO BLOCO F05 NO EIXO X	49
5.2 MODELAGEM DA SÉRIE DO DESLOCAMENTO DO BLOCO F05 NO EIXO Y	51
5.3 MODELAGEM DA SÉRIE DO DESLOCAMENTO DO BLOCO F13 NO EIXO X	53
5.4 MODELAGEM DA SÉRIE DO DESLOCAMENTO DO BLOCO F13 NO EIXO Y	55
5.5 MODELADO DA SÉRIE DO DESLOCAMENTO DO BLOCO F19 NO EIXO X...	57
5.6 MODELADO DA SÉRIE DO DESLOCAMENTO DO BLOCO F19 NO EIXO Y...	59
<b>6 CONCLUSÕES</b> .....	<b>62</b>
6.1 RECOMENDAÇÕES PARA TRABALHOS FUTUROS .....	62
<b>REFERÊNCIAS</b> .....	<b>63</b>
<b>ANEXO 1 – ALGORITMOS UTILIZADOS NAS REDES NEURAIS</b> .....	<b>66</b>

<b>ANEXO 2 – FUNÇÕES DE ATIVAÇÃO.....</b>	<b>71</b>
<b>ANEXO 3 – CÁLCULOS ESTATÍSTICOS E MATEMÁTICOS.....</b>	<b>72</b>
<b>APÊNDICE 1 – CÓDIGO IMPLEMENTADO .....</b>	<b>73</b>

## 1 INTRODUÇÃO

A fim de evitar perdas materiais e humanas, dentro e nas periferias de uma barragem, torna-se necessário verificar a saúde estrutural da mesma.

Os desastres das barragens de Austin citados por Rich (2006) e a de Canyon Lake citada por Graham (1999) são exemplos do risco que representam as falhas em barragens. Diante disso, os controles de segurança são fundamentais para garantir a funcionalidade e integridade estrutural da usina.

O monitoramento dos mesmos e dado de acordo com as propriedades estruturais da barragem que podem ser divididas em dois tipos (P. BUKENYA, 2014):

**Propriedades Estáticas:** são as características medidas como deformações e tensões na barragem produtos de fatores ambientais como velocidade do vento, temperatura ambiental e temperatura da água.

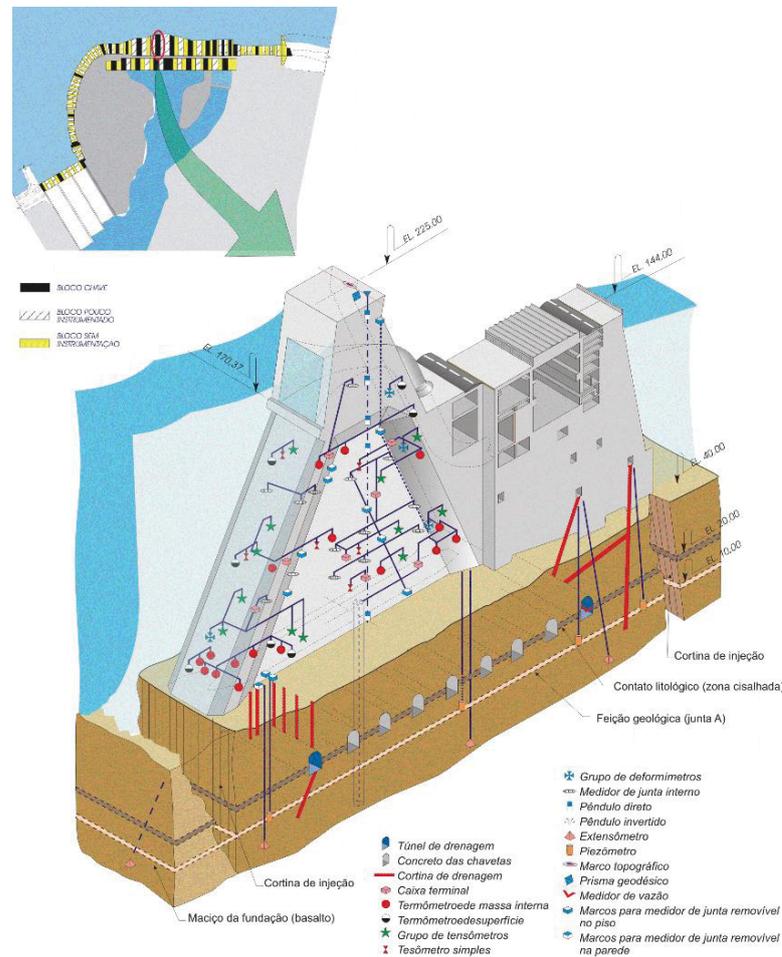
**Propriedades Dinâmicas:** aquelas associadas às propriedades dinâmicas da barragem (frequência natural, relação de amortecimento, etc) são extraídas da aceleração, usando técnicas de análise modal.

No monitoramento estático, os dados obtidos dos sensores são analisados utilizando modelos estatísticos ou determinísticos. Os modelos estatísticos estão fundamentados em correlações entre as variáveis ambientais (nível da água, temperatura do ambiente, velocidade do vento) e a resposta da barragem (deslocamentos, pressão e tensões), apresentando uma relação entre o comportamento presente e o passado da barragem. No entanto, os modelos determinísticos estabelecem uma relação entre a carga e a resposta da barragem por meio de uma análise estrutural (BIANCHI; BREMEN, 2001).

Existem aproximadamente 2400 instrumentos e mais de 5000 drenos que, continuamente, monitoram o estado da barragem da Usina Hidrelétrica da Itaipu Binacional e formam uma base de dados de mais de 30 anos (BINACIONAL, 2018). Na FIGURA 1, pode-se observar a instrumentação da barragem. Os modelos construídos a partir dos dados obtidos destes instrumentos são usados para prever o comportamento da barragem e detectar se existe alguma anormalidade com relação ao comportamento típico da mesma e, assim, atuar preventivamente se for necessário. Um grupo importante desses dados coletados são mensurados pelo instrumento chamado pêndulo direto que mede o movimento horizontal da crista da barragem com relação a fundação da estrutura. Segundo Silveira (2013) o dito

instrumento constitui um dos mais robustos e precisos para a medição de deslocamentos horizontais em barragens de concreto de altura mínima de 30 metros.

FIGURA 1 INSTRUMENTAÇÃO DA BARRAGEM



FONTE: ITAIPU (2018).

Pequenos deslizamentos horizontais podem gerar fissuras que podem comprometer a estrutura toda. Portanto, o estudo do monitoramento e previsão das séries temporais associadas a esses deslocamentos se torna necessário para aumentar a segurança e a execução de uma correta manutenção preventiva (BIANCHI; BREMEN, 2001).

## 1.1 OBJETIVOS

### 1.1.1 Objetivo geral

Implementar redes neurais recorrentes (RNR) do tipo *Long Short Term Memory* (LSTM) para gerar um mês de previsões das séries temporais de deslocamentos horizontais dos blocos-chave F05, F13 e F19 no eixo X e Y da barragem principal da usina hidrelétrica de Itaipu.

### 1.1.2 Objetivos específicos

- Analisar as séries temporais dos deslocamentos dos blocos chaves F05, F13 e F19
- Comparar a eficiência do modelo obtido com modelos consagrados na literatura como o *Autoregressive Integrated Moving Average* (ARIMA) e as redes neurais artificiais (RNA) do tipo *Multilayer Perceptron* (MLP).
- Fazer uma busca por grade dos hiperparâmetros básicos para as RNA do tipo MLP e a RNR do tipo LSTM.

## 1.2 JUSTIFICATIVA

A compreensão da sazonalidade e ciclos juntamente com a análise gráfica bem como o uso de estatísticas como média, variância, amplitude, máximos e mínimos históricos não são suficientes para se criar planos de ações ou referencias para tomada de decisão. Neste trabalho é proposta a implementação de RNR do tipo LSTM para a previsão das séries temporais dos deslocamentos dos blocos F05, F13, é F19 nos eixos X e Y, demonstrando assim sua capacidade para mapear as estruturas de autodependência linear e não linear presentes na série.

As projeções obtidas têm como objetivo apresentar uma ferramenta auxiliar à equipe de engenheiros bem como à equipe técnica, tendo em vista o aprimoramento nas tomadas de decisões referentes ao planejamento de possíveis medidas corretivas.

## 2 REVISÃO DE LITERATURA

Na área de previsão de séries temporais a ideia básica é obter e analisar as observações históricas para determinar um modelo que capture o processo interno que gera tais dados. Logo então, extrapola-se os dados para se obter previsões de valores futuros. Por muito tempo esta área esteve totalmente dominada por os métodos tradicionais lineares estatísticos, mas os ditos modelos têm uma desvantagem; não podem capturar relações não lineares dos dados. Em contrapartida, segundo Zhang (2012) as redes neurais se apresentam como uma alternativa bastante promissora para a área, pois sua estrutura não linear facilita a captura das propriedades mais complexas do modelo real. Também, por ser um método orientado por dados, não há o problema de erro de especificação dos modelos paramétricos, e está demonstrado que ao ter uma função de ativação não linear ele se comporta como um aproximador universal (CYBENKO, 1989).

Na literatura atual, o uso de redes neurais está bastante estendido na área de previsões das séries temporais. Em Corrêa (2015) um modelo híbrido é composto pelo método estatístico não linear do tipo *Autoregressive conditional heteroskedasticity* (GARCH) e o linear ARIMA com a transformada wavelet e a combinação linear com uma rede MLP é utilizada para modelar a série temporal de deslocamentos dos blocos da usina Itaipu com excelentes resultados.

Em Bandara (2017) os autores propõem redes neurais recorrentes do tipo LSTM para previsão em sistemas dinâmicos caóticos e testam a proposta em séries temporais obtidas com o sistema Lorenz 96, a equação Kuramoto-Sivashinsky e um protótipo de modelo climatológico, tendo resultados satisfatórios.

Em Mata (2011) o autor compara o uso de múltiplos modelos de regressão linear e uma rede neural do tipo MLP para a previsão dos deslocamentos dos blocos da barragem de Alto Rabagão, localizada em Portugal, e faz a interpretação do comportamento dos blocos de concreto e, de acordo a isso, conclui que a capacidade das redes neurais de modelar a série em condições de temperatura extrema é maior e que mostra maior flexibilidade que o uso dos modelos de regressão. Finaliza com que ambos modelos podem ser combinados para uma melhor análise do estado da barragem.

Também é possível notar o crescente uso de redes recorrentes na previsão das séries temporais, por exemplo em Ma et al. (2015) a rede recorrente do tipo LSTM

foi testada com vários métodos de predição de velocidade de trafico incluindo SVM, filtro de kalman, o método estatístico ARIMA, entre outros, tendo o melhor desempenho a rede neural LSTM.

Em Hsieh (2011) um modelo com a transformada wavelet e o uso de redes neurais recorrentes baseados no algoritmo de colonização de formigas é utilizado para realizar a predição dos preços das ações dos valores do mercado da bolsa. Assim também se tem usos na predição próxima do clima, no setor da predição da carga energética doméstica, predição da velocidade do vento, entre outros, em todos eles os algoritmos recorrentes demonstraram ser eficazes na predição (SHI et al., 2015; SHI; XU; LI, 2018; LIU; MI; LI, 2018).

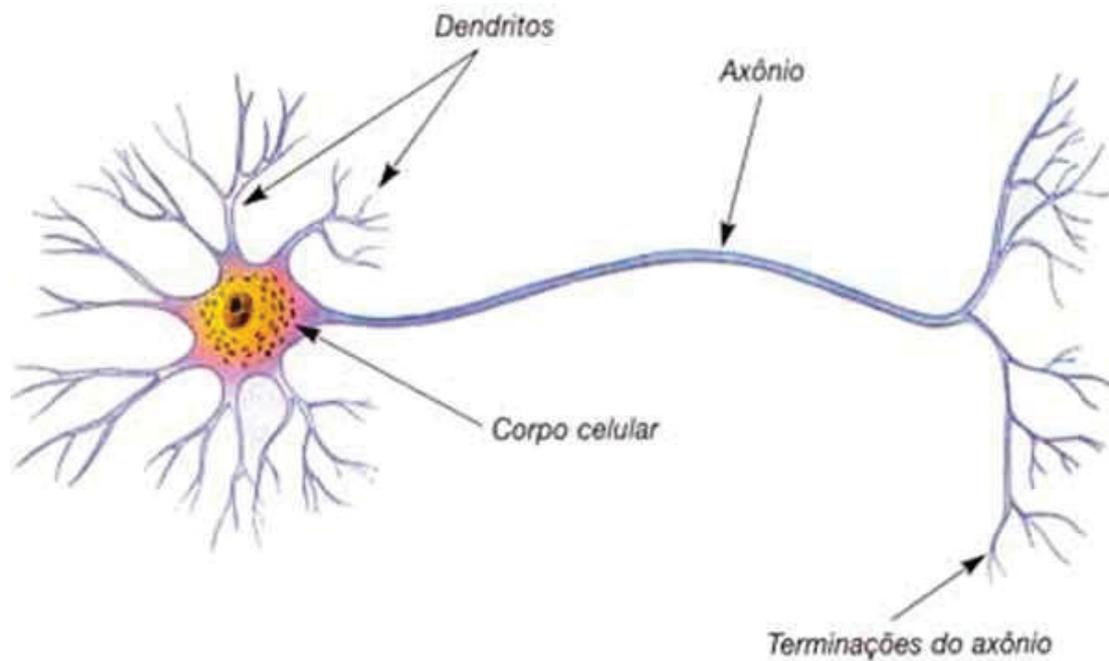
## 2.1 REDES NEURAIS ARTIFICIAIS (RNA)

As redes neurais artificias (RNA) foram baseadas em parte no sistema nervoso central biológico que é composto de milhares de unidades simples de neurônios interconectados numa rede densa. Estima-se que o cérebro humano tem uma rede interconectada de aproximadamente  $10^{11}$  neurônios conectados em média a aproximadamente  $10^4$  outros neurônios e as mudanças de estado destes neurônios levam em media  $10^{-1}$  segundos em acontecer, muito mais que os  $10^{-10}$  segundos que leva um computador para executar a mesma tarefa.

Esta observação estabeleceu a ideia de que as habilidades de processamentos da informação dos sistemas neurais biológicos devem estar ligadas a processamentos altamente paralelos (MITCHELL, 1997). Uma das motivações mais importante do desenvolvimento das RNA é captar este tipo de processamentos.

Na FIGURA 2 é mostrada a esquematização simplificada de um neurônio biológico, este é constituído por um corpo celular (soma) que contém o núcleo da célula, diversos dendritos através dos quais os impulsos elétricos são recebidos e um axônio por meio do qual são enviados impulsos elétricos.

FIGURA 2 ESQUEMA SIMPLIFICADO DE UM NEURÔNIO



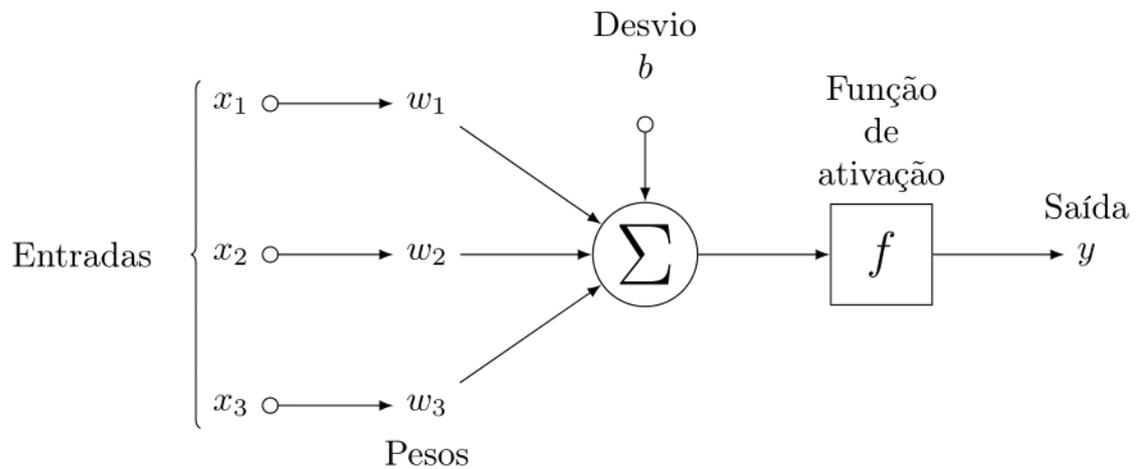
FONTE: HERMANO (2009).

A analogia de uma RNA a partir desta figura é a seguinte:

- A conexão entre os neurônios é representada pelos pesos sinápticos  $w_j^k$ , função análoga a do axônio. Sendo  $j$  e  $k$  o índice do neurônio  $j$  da camada  $k$ .
- O somatório  $\Sigma$  e a função de ativação representam o corpo celular e a ação potencial nos dendritos.
- Os dendritos que representam a entrada ao neurônio.

O perceptron simples de uma camada, observado na FIGURA 3, pode ser comparado com a FIGURA 2 para se observar as semelhanças entre o modelo simplificado de um neurônio e uma RNA.

FIGURA 3 PERCEPTRON SIMPLES DE UMA CAMADA



FONTE: AUTOR (2018).

Para explicar melhor o funcionamento de uma rede neural artificial tem-se que compreender o funcionamento do neurônio na mesma, o neurônio é a peça fundamental do cérebro e da rede em si, sua função é receber informações de outros neurônios e passar a informação para as outras células. Este traspasso de informação é representado no esquema simplificado pela conexão entre os neurônios e adota o nome de axônio, numa RNA isto é representado por meio dos pesos sinápticos  $w_j^k$ .

Assim como as conexões se intensificam na medida que essa é utilizada e logo são somadas no corpo da célula e levada a outros neurônios, na RNA os pesos sinápticos formam uma combinação linear com os neurônios e são somadas para logo passar para outra camada e repetir o processo, isto é facilmente exemplificado na FIGURA 4. Matematicamente, isto é expresso a seguir para uma rede perceptron de uma capa (SLP):

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, W = \begin{bmatrix} w_1^1 \\ w_2^1 \\ \vdots \\ w_n^1 \end{bmatrix}, b \quad (1)$$

$$y = f(W^T X + b) \quad (2)$$

Onde  $X$  representa os vetores de entrada,  $W$  os pesos sinápticos,  $b$  o desvio e finalmente  $f$  a função de ativação. Para uma MLP como na FIGURA 4 pode ser generalizada para a seguinte fórmula:

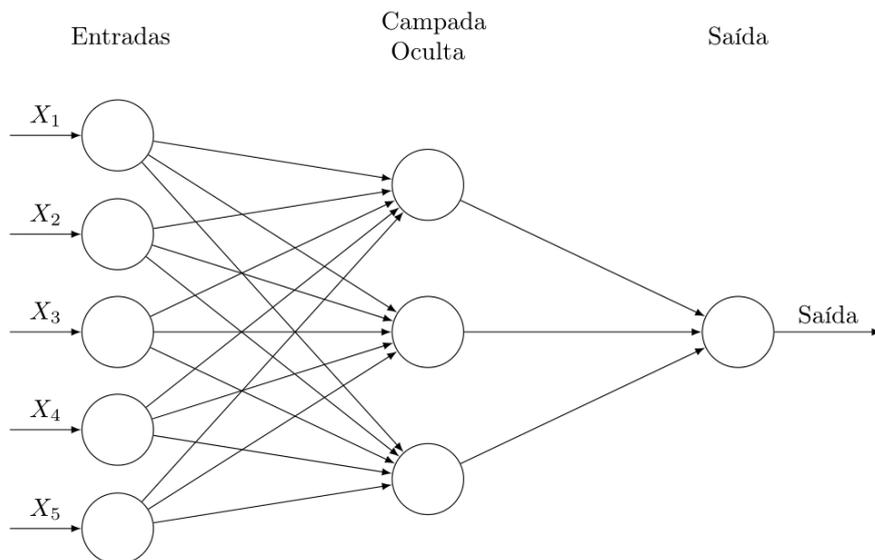
$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{p_0} \end{bmatrix}, W^l = \begin{bmatrix} w_{11}^l \\ w_{12}^l \\ \vdots \\ w_{p_{l-1}p_l}^l \end{bmatrix}, a^l = \begin{bmatrix} a_1^l \\ a_2^l \\ \vdots \\ a_{p_l}^l \end{bmatrix}, b^l = \begin{bmatrix} b_1^l \\ b_2^l \\ \vdots \\ b_{p_l}^l \end{bmatrix}, Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{p_l} \end{bmatrix} \quad (3)$$

Sendo  $X$  as entradas,  $W^l$  os pesos sinápticos da camada  $l$ ,  $a^l$  a saída da função de ativação  $f$  para cada elemento da camada  $l$ ,  $b^l$  o desvio para cada neurônio da camada  $l$  e  $Y$  sendo o vetor de saída com  $p_j$  elementos. Por tanto sendo  $a^0 = X$  teríamos os seguintes valores de  $a$ .

$$a^l = f_l((W^l)^T a^{l-1} + b^l) \quad (4)$$

Sendo  $l > 0$ ,  $f_l$  a função de ativação da camada  $l$  e finalmente definimos  $L$  como a última camada, portanto  $a^L = Y$ .

FIGURA 4 PERCEPTRON DE MÚLTIPLAS CAMADAS (MLP)



FONTE: AUTOR (2018).

### 2.1.1 Características de uma RNA

Uma RNA está definida principalmente por dois componentes fundamentais: a função de ativação das camadas e a topologia ou arquitetura da rede. De acordo a arquitetura pode ser definida por dois tipos (HAYKIN, 2009):

- Diretas (*Feedforward*): as conexões vão de uma camada a outra seguinte em sentido direto, os neurônios não apresentam interconexão com as camadas anteriores e entre os da mesma camada, somente com as das seguintes camadas. A FIGURA 2 e a FIGURA 4 são exemplos deste tipo de redes.
- Recorrentes (*Feedback*): são aquelas que possuem realimentação das saídas para as entradas. Um exemplo deste tipo de rede e a rede neural do tipo LSTM, sendo utilizado bastante na área de modelagem de sequências e atualmente na previsão das séries temporais (BANDARA; BERGMEIR; SMYL, 2017).

Com relação a função de ativação as seguintes propriedades são desejadas:

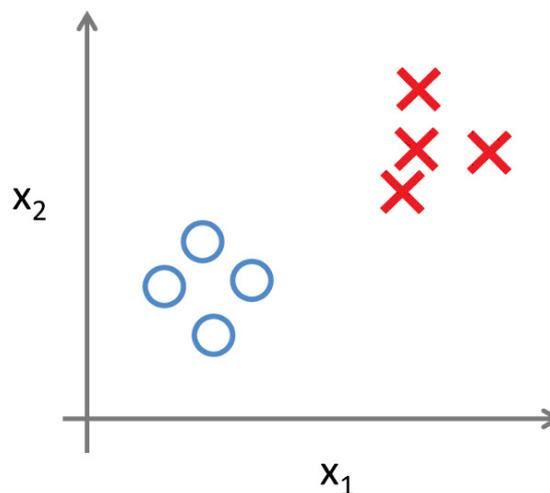
- Não linearidade: deve ser não linear para assim poder ser uma aproximador universal de funções contínuas num conjunto compacto de  $R^n$  (CYBENKO, 1989).
- Diferenciação contínua: esta propriedade é desejada devido a utilização de métodos baseados no gradiente para a otimização e treinamento da rede (SNYMAN, 2005).
- Saídas limitadas: quando a saída da função é finita se tem que o treinamento por métodos de gradiente e mais estável (LIEW; KHALIL-HANI; BAKHTERI, 2016).
- Monotônica: quando a função é monótona no modelo com uma camada é garantido que a superfície do erro seja convexa, facilitando assim o treinamento da rede (WU, 2009).

### 2.1.2 Treinamento de Redes Neurais

Citando a Haykin e Engel (2001) tem-se que o treinamento das redes neurais pode ser dividido em dois tipos:

- **Treinamento Supervisionado:** O treinamento é dado pelo ajuste numérico dos pesos sinápticos  $W_{kj}^l$ , que visam a maximização ou minimização de uma função objetivo. No caso da RNA, geralmente é dado a partir da minimização do erro quadrático médio entre os valores gerados na saída da RNA e os valores reais. O *Backpropagation* é o algoritmo que calcula a gradiente dos pesos sinápticos que logo e combinada com uma técnica de otimização para ajustar os valores dos pesos sinápticos, minimizando a função objetivo.

FIGURA 5 EXEMPLO DE TREINAMENTO SUPERVISIONADO

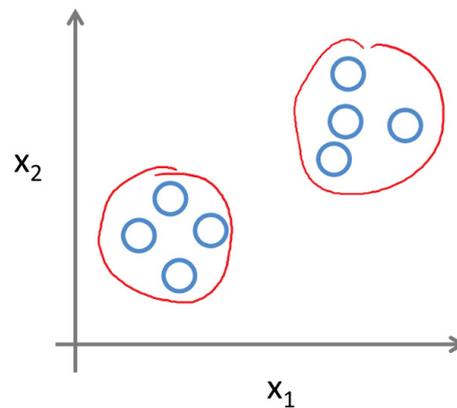


FONTE: LAKSHAY (2017)

- **Treinamento Não Supervisionado:** Neste caso, a característica principal é a ausência de algum elemento externo supervisor, ou seja, não há valores reais para ser utilizados como padrões na saída, tomando em contrapartida as propriedades estatísticas do conjunto de dados usados

no treinamento. Segundo Palit e Popovic (2005) é principalmente utilizado na classificação de dados.

FIGURA 6 EXEMPLO DE TREINAMENTO NÃO SUPERVISIONADO

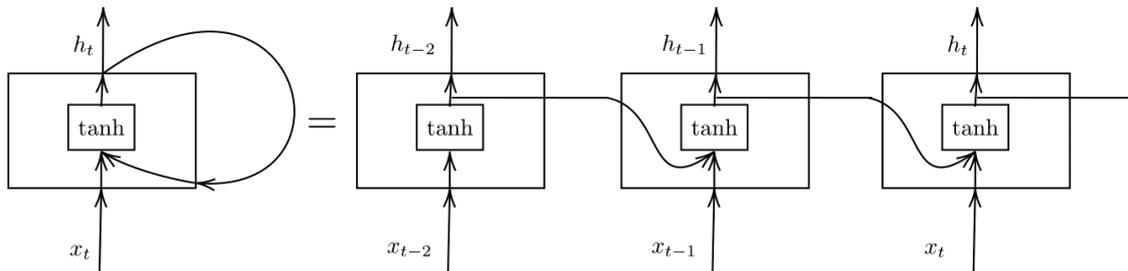


FONTE: LAKSHAY (2017)

### 2.1.3 Redes Neurais Recorrentes

As redes neurais recorrentes levam esse nome devido a topologia que utilizam, são recorrentes no sentido de que a informação previamente computada nos estados do neurônio da rede é usada na iteração seguinte. Esse procedimento é o mesmo para cada vetor da sequência de vetores de entrada. Historicamente não foram muito utilizadas devido ao custo computacional do treinamento produzido pelo processo de calcular os valores anteriores e considerar os estados em cada iteração, mas atualmente é um dos tipos de redes mais utilizadas para trabalhar com sequências devido ao avanço computacional (BUDUMA; LOCASCIO, 2017).

FIGURA 7 REDE NEURAL RECORRENTE BÁSICA

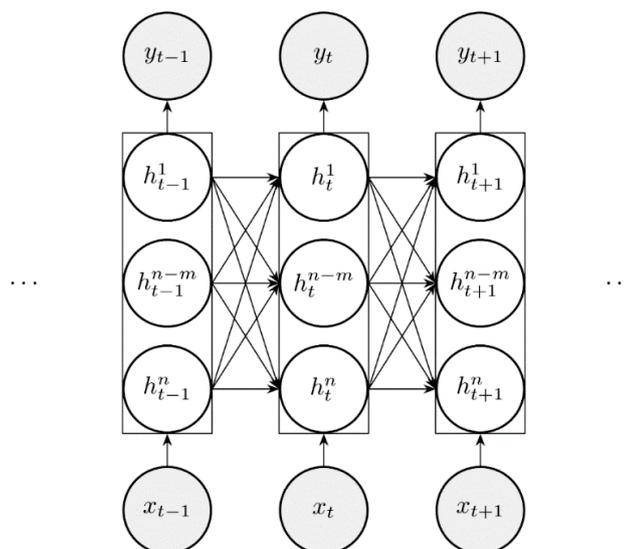


FONTE: AUTOR (2018).

Na FIGURA 7, pode-se observar uma rede neural recorrente desdobrada. Nesta, as entradas  $x_t$  geram as saídas  $h_t$  que logo são realimentadas na rede para serem utilizadas na seguinte entrada  $x_{t+1}$ . Cada passo do tempo gera uma nova camada temporal denominada célula no que seria uma arquitetura muitos para muitos.

Na FIGURA 8, pode-se observar claramente como funcionam estas células, internamente elas incorporam os neurônios  $h_t^j$  que são conectados aos valores do vetor temporal  $x_t$ , onde se o vetor é multivariável contém todos os valores das variáveis espaciais nesse exato momento, tendo cada um conexões com os neurônios desse tempo e passando esses valores a seguinte célula temporal.

FIGURA 8 RNR DE MÚLTIPLAS CÉLULAS E MÚLTIPLOS NEURÔNIOS

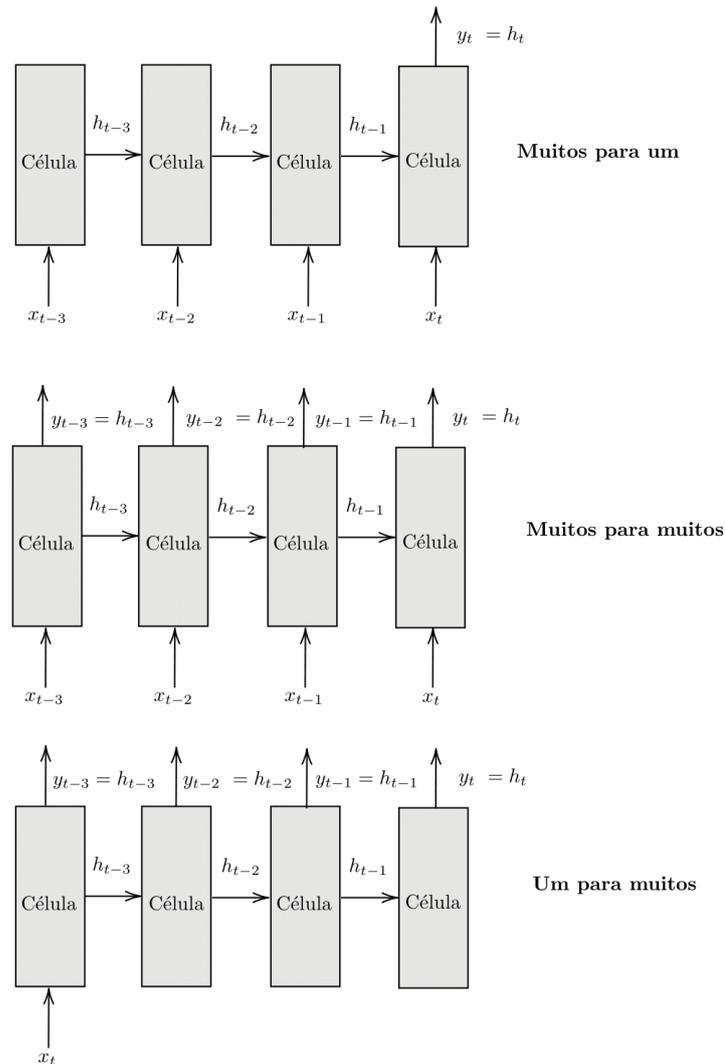


FONTE: AUTOR (2018).

De acordo a Karpathy (2015) as redes neurais recorrentes podem ser classificadas de acordo a seguintes arquiteturas observadas na FIGURA 9:

- Um para muitos: um valor que gera uma sequência, a realimentação das células se faz de forma direta sem considerar a quantidade de passos de tempo e todas as saídas em cada célula temporal são consideradas. Exemplo: Uma imagem que gera um conjunto de palavras.
- Muitos para um: nesta arquitetura a saída é somente dada pela última célula temporal e neste caso e considerado todos os passos temporais para a geração deste vetor final. Exemplo: Análises de sentimento numa rede social dado uma frase.
- Muitos para muitos: neste caso se considera tanto o passo do tempo nos vetores de entrada e as saídas do mesmo, se tem uma sequência de valores que gera outra sequência de valores. Exemplo: Análises e rotulação de quadros de vídeo.

FIGURA 9 ARQUITETURAS BÁSICAS DAS REDES NEURAIS RECORRENTES



FONTE: AUTOR (2018).

#### 2.1.4 O Problema da fuga de gradientes

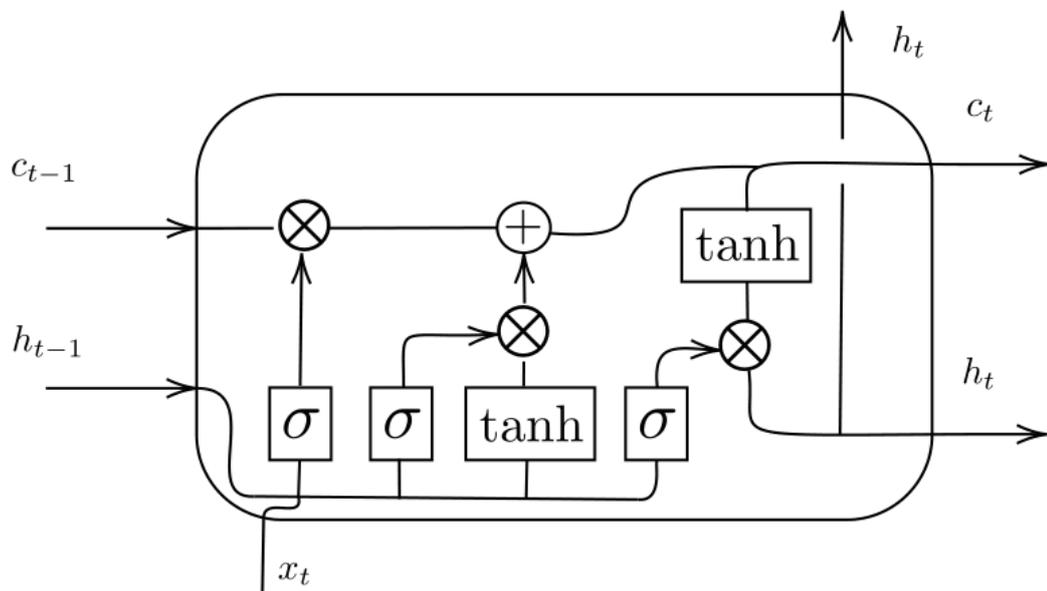
O treinamento das redes neurais recorrentes é dado pelo uso de uma variação do algoritmo de *backpropagation* denominado *Backpropagation Through Time* (BPTT) que aplica o conceito do algoritmo através da dimensão temporal da rede recorrente, ou seja, a derivada dos tempos dos estados anteriores é calculada para determinar a variação do erro, atualizando assim todas as camadas e células da rede. A causa da elevada dimensionalidade da rede, as derivadas dos erros da rede sofrem de um problema chamado de fuga de gradientes (*Vanishing Gradient*) com este algoritmo, o qual é causado pela aplicação da regra da cadeia, deixando assim o gradiente muito

pequeno ou muito grande evitando uma modificação adequada dos valores dos pesos sinápticos. Um tipo de rede recorrente que soluciona este tipo de inconveniente é a denominada rede do tipo LSTM, que adere uma estrutura de comportas que define quanto de cada dado será utilizado na iteração seguinte facilitando assim o uso do BPTT no treinamento (BUDUMA; LOCASCIO, 2017).

### 2.1.5 Redes do tipo LSTM

Apresentadas por Hochreiter e Schmidhuber (1997) as redes recorrentes do tipos LSTM apresentaram uma resposta a problemática da fuga de gradientes utilizando uma estrutura de comportas com memória. Assim, a informação mais importante é retida em vários passos de tempo e aquela que não representa mais a variação atual é esquecida. Com isso o modelo LSTM evita que o cálculo do gradiente tenha um comportamento mais estável.

FIGURA 10 ESTRUTURA DE UMA UNIDADE LSTM



FONTE: AUTOR (2018).

Na FIGURA 10 é possível observar a estrutura de comportas de uma unidade LSTM e dentro desta estrutura onde \* representa a operação elemento a elemento elas cumprem as seguintes funções:

- Comportas de esquecimento: estabelece que tanto do valor anterior será usado para atualizar o estado da célula e a saída da mesma. A equação da comporta e a seguinte  $for_t = \sigma((W_{for})^T \begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix} + b_{for})$ .
- Comportas de entrada: define quanto do valor da entrada será utilizada para atualizar o estado da célula e a saída da mesma. A equação da comporta é a seguinte  $in_t = \sigma((W_{in})^T \begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix} + b_{in})$ . Junto com a comporta de esquecimento são as encarregadas de atualizar o estado da célula.
- Estado da célula: define o estado interno da célula dada uma entrada temporal dos dados. Está definida pela seguinte função  $c_t = for_t * c_{t-1} + in_t * \tilde{c}$  onde  $\tilde{c} = \tanh\left((W_c)^T \begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix} + b_c\right)$  e  $c_{t-1}$  é o estado anterior da célula.
- Comporta de saída: expõe o conteúdo da célula na saída. A função de saída corresponde a seguinte  $h_t = out_t * \tanh(c_t)$  e  $out_t = ((W_{out})^T \begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix} + b_{out})$ .

### 3 METODOLOGIA

Para a elaboração deste trabalho se adotou a seguinte metodologia: Será proposta uma arquitetura para a implementação das redes neurais e serão testadas com as séries temporais de deslocamento dos blocos-chave F05, F13 e F19 obtidos dos pêndulos diretos instalados na barragem e, finalmente, para demonstrar a sua eficácia com ditas séries a mesma será comparada com métodos clássicos como a RNA do tipo MLP e o método estatístico ARIMA.

O monitoramento e análise das propriedades dinâmicas da barragem são baseadas principalmente no comportamento da barragem perante a vibração e ficam fora do escopo deste trabalho.

No QUADRO 1, pode-se observar a unidade e a orientação das medidas obtidas nos blocos-chave mencionados, com a ideia de implementar um modelo de previsão com a utilização de Redes Neurais Recorrentes do tipo LSTM com um horizonte de previsão de 30 dias.

QUADRO 1 SÉRIES DE DESLOCAMENTO DOS BLOCOS-CHAVE DA BARRAGEM PRINCIPAL DA USINA HIDRELÉTRICA DE ITAIPU.

SÉRIE	AFERIÇÃO	UNIDADE DE MEDIDA
F05x	Deslocamento do bloco F05 sentido longitudinal ao fluxo do rio	mm
F05y	Deslocamento do bloco F05 sentido transversal ao fluxo do rio	mm
F13x	Deslocamento do bloco F13 sentido longitudinal ao fluxo do rio	mm
F13y	Deslocamento do bloco F13 sentido transversal ao fluxo do rio	mm

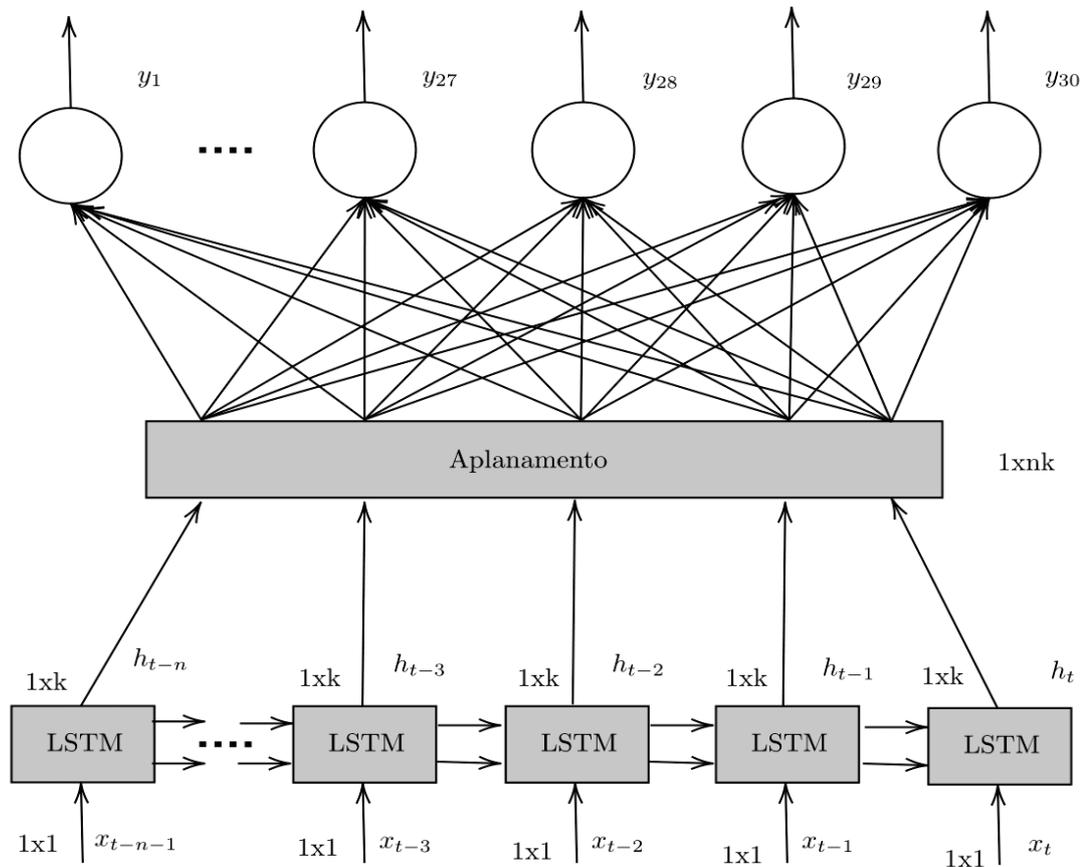
F19x	Deslocamento do bloco F19 sentido longitudinal ao fluxo do rio	mm
F19y	Deslocamento do bloco F19 sentido transversal ao fluxo do rio	mm

FONTE: CORRÊA (2015).

Para tal efeito, apresenta-se a estrutura proposta na FIGURA 11 baseado em THOMAS (2018). Em esta pode se ver uma estrutura do tipo muitos para muitos seguido por um modulo de concatenação para finalmente obter as 30 saídas através de uma camada perceptron com uma função de ativação linear associado a estas. Cada célula LSTM da rede gera uma saída de k elementos que logo são concatenadas para formar um vetor de 1xnk dimensões, a estrutura proposta aproveita a recorrência da rede no sentido em que cada dado temporal é processado sequencialmente e logo utilizado pela capa de saída para determinar cada saída de acordo a uma combinação linear do tipo  $Y = (W_k)^T h_s + b_s$  sendo Y a saída vetorial que representam os 30 valores com uma função de ativação linear,  $h_s$  a saída concatenada dos módulos e  $W_k$  a matriz dos pesos sinápticos para o modulo de aplanamento.

A metodologia para a escolha dos melhores hiperparâmetros n e k para a estrutura proposta será feita através de uma busca por grade especificada em Buduma e Locascio (2017), onde diversos hiperparâmetros serão testados um a um utilizando dados de validação para logo serem testados com dados fora da amostra assim como é recomendado por Armstrong (2007). Para provar a eficiência da estrutura na predição a mesma será comparada com uma rede MLP de uma camada oculta que de acordo com Funahashi (1989) é suficiente para modelar uma função com mapeamento contínuo otimamente e já foi testado com sucesso para estas séries temporais em Corrêa (2015). Também foi testada contra o método estatístico ARIMA para observar a diferença com modelos tradicionais.

FIGURA 11 ESTRUTURA PROPOSTA



FONTE: AUTOR (2018).

No QUADRO 2 se mostram todos os hiperparâmetros a serem testados nos modelos. A função objetivo é definida com relação ao erro quadrático médio (MSE) sendo a mesma definida por  $C(x) = \sum_{n=1}^m \sum_{p=1}^{30} (y_{pn}(x) - o_{pn})^2$  onde  $m$  é a quantidade de dados no lote da época de treinamento,  $y(x)_{pn}$  a predição obtida em base ao dado  $n$  e  $o_{pn}$  o valor real na saída  $p$ . Como o sobre treinamento do modelo em base aos dados de treinamento é um problema comum dentro das redes neurais, um modo de lidar com isso é o uso de técnicas de regularização baseadas em penalidades aplicadas a os pesos da rede e definidas de acordo as métricas  $L^1$  e  $L^2$  definidas no espaço  $L^P$ , com  $L^1$  igual à  $\lambda_1 \sum |w|$  e a  $L^2$  igual à  $\lambda_2 \sum w^2$  sendo  $w$  a representação dos pesos sinápticos da rede. Neste trabalho foram utilizados ambos devido as propriedades que cada regularização contribui no treinamento da rede; a  $L^2$  estabiliza os pesos das variáveis altamente correlacionadas minimizando assim o grupo de

variáveis que aportam ao modelo e a  $L^1$  reduz o ruído no mesmo (GOODFELLOW; BENGIO; COURVILLE, 2016). Considerando isso, o custo final fica da forma  $\mathcal{C}(x, w) = \sum_{n=1}^m \sum_{p=1}^{30} (y_{pn}(x) - o_{pn})^2 + \lambda_1 \sum |w| + \lambda_2 \sum w^2$ . Para o treinamento do tipo supervisionado foi utilizado o otimizador Adam que está baseado no algoritmo do gradiente descendente com o agregado que atualiza os valores do gradiente tomando em conta a variação nos momentos de segunda ordem dos erros (KINGMA; BA, 2014). Finalmente as séries temporais foram normalizadas utilizando a fórmula  $\frac{x - \min(x)}{\max(x) - \min(x)}$  em um intervalo de  $[0, 1]$  sendo somente utilizadas uma só sequência por lote ( $m=1$ ) o que representa o uso do algoritmo gradiente descendente estocástico (SGD) dentro do otimizador.

QUADRO 2 HIPERPARÂMETROS DOS MODELOS

Neurônios	Entradas	Saída
5,10,15,20,25,30	5,10,15,20,25,30	30
5,10,15,20,25,30	5,10,15,20,25,30	30
Ativação da Cam. Oculta	Regularização	Cam. Ocultas
Tanh	$(L^1 \text{ e } L^2) 0.5e-16$	1
Tanh	$(L^1 \text{ e } L^2) 0.5e-16$	1
Erro	Otimizador	Rede
MSE	Adam	LSTM
MSE	Adam	MLP

FONTE: AUTOR (2018).

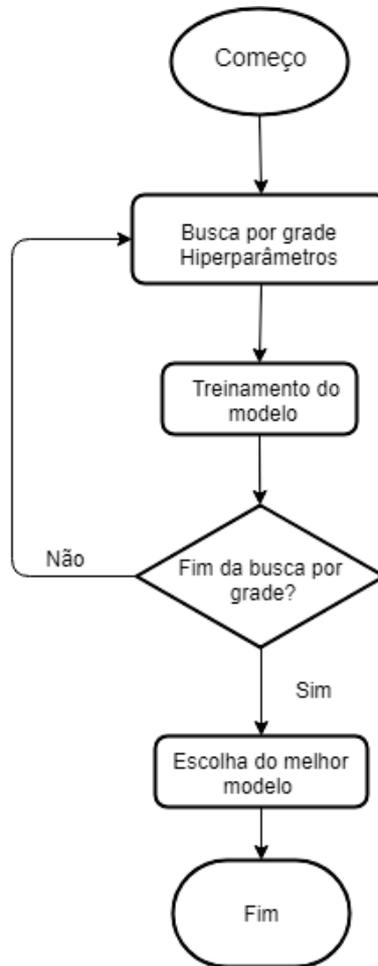
Para a análise das séries temporais foram mostrados os últimos 600 dados das amostras e estão marcados quais são os dados de validação e teste, os dados de validação e teste são os últimos 60 dados da série temporal, sendo divididos em duas partes iguais para cada. Finalmente, algumas estatísticas descritivas foram comentadas para cada uma das séries temporais. Também serão apresentados os gráficos da função de autocorrelação total (FAC) e parcial (FACP) das séries temporais com um intervalo de confiança de 95% marcado com a cor azul no mesmo (PARZEN, 1963).

Os resultados dos modelos das séries e as previsões feitas pelos melhores modelos escolhidos segundo a metodologia descrita anteriormente serão mostrados de acordo a seguintes métricas: o erro quadrático médio (MSE), o erro absoluto médio (MAE) e finalmente o erro absoluto porcentual médio (MAPE). Na configuração o parâmetro E determina a quantidade de dados estabelecidos na sequência de entrada, N a quantidade de neurônios e S a sequência de saída, com relação ao método ARIMA, AR representa o modelo autoregressivo do método, I a diferenciação da série e MA o modelo por medias moveis que combinados formam o modelo completo.

A implementação foi feita em Python utilizando a biblioteca (*package*) *Keras* criada por Chollet (2015), a biblioteca para aprendizagem de máquina utilizado por o *Keras* neste trabalho e o *Tensorflow* de Abadi et al. (2016), esta facilita o treinamento de redes neurais através da utilização de cálculo tensorial que facilitam a paralelização do código no mesmo. Os algoritmos de BPTT e BP estão completamente implementados dentro desta biblioteca. Para a implementação da modelagem ARIMA foi utilizado o software *STATGRAPHICS* com o método de obtenção automática de parâmetros baixo o critério de informação Akaike.

Na FIGURA 12 é mostrado o fluxograma seguido para o treinamento e escolha dos modelos neurais, nela primeiramente se estabelecem os hiperparâmetros para o treinamento segundo o QUADRO 2, logo o mesmo é treinado com o *Keras* em base a os dados de treinamento. Uma vez finalizado o treinamento e calculado os erros de previsão dos dados de validação e se continua para treinar o modelo seguinte até finalizar a busca por grade. Após terminado, é escolhido o modelo que tem o menor erro de validação para logo ser verificado com os dados de teste. No treinamento o erro quadrático médio (MSE) e utilizado como função objetivo e se estabelece um critério de parada que segue a seguinte regra, verifica-se que em cada iteração há uma melhoria de pelo menos 1% com relação ao menor erro obtido se esse não for o caso se espera 10 iterações até que a dita melhora ocorrer, se mesmo assim a melhora não ocorrer se dá por terminado o treinamento. A máquina onde foi feito o treinamento tem um processador Intel I7 4790 de 8 núcleos com 3.6 GHz de velocidade e uma memória RAM de 16GB do tipo DDR3.

FIGURA 12 FLUXOGRAMA DO TREINAMENTO E ESCOLHA DOS MODELOS NEURAIIS



FONTE: AUTOR (2018).

## 4 ANÁLISE DAS SÉRIES TEMPORAIS

### 4.1 SÉRIE TEMPORAL DO DESLOCAMENTO DO BLOCO F05 NO EIXO X

No GRÁFICO 1, pode-se observar os últimos 600 dados dos 1924 da série temporal do deslocamento do bloco F05 no eixo X da barragem. É possível observar quais serão os dados utilizados para a validação e para o teste, também foi feita a análise da função de autocorrelação e a autocorrelação parcial que podem ser observadas graficamente no GRÁFICO 2. A série temporal apresenta autocorrelação total do tipo senóide amortecido e dois picos na autocorrelação parcial. Com isso pode-se dizer que a série temporal possui uma parte sistemática alta do tipo autoregressivo devido a que elas estão fora do intervalo de confiança, por tanto pode ser modelada linearmente e um modelo ARIMA seria capaz de captar boa parte do comportamento da série.

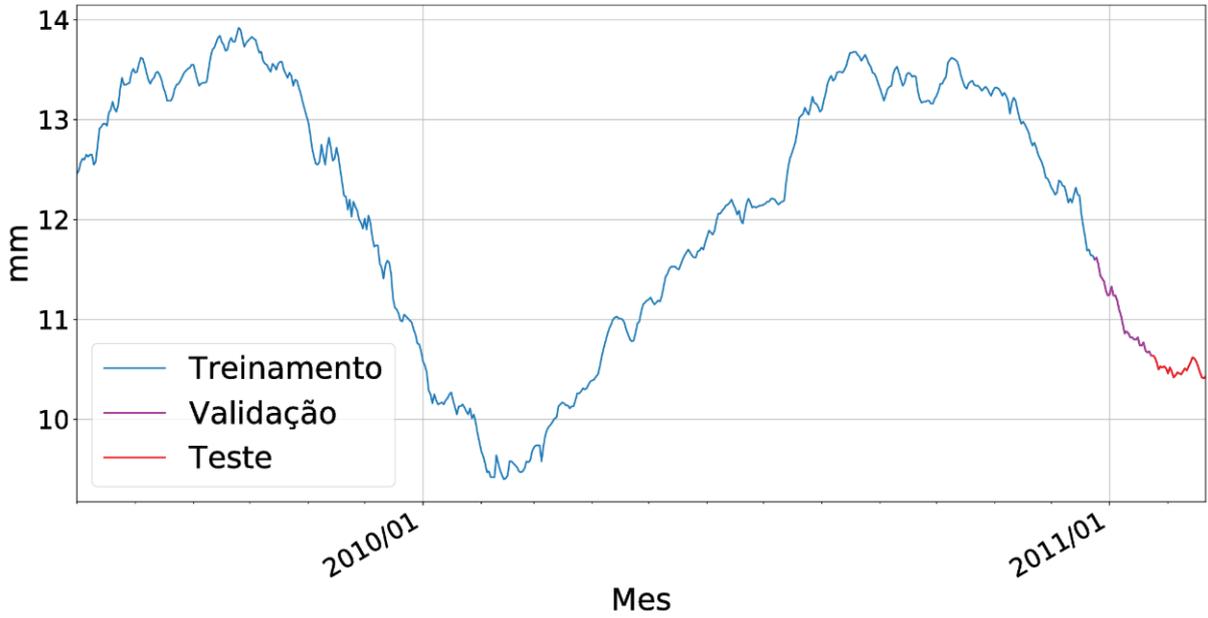
TABELA 1 DADOS DESCRITIVOS DA SÉRIE TEMPORAL F05X

MEDIDA	RESULTADO
<b>MEDIA</b>	11.3265
<b>MIN</b>	8.93
<b>MAX</b>	13.92
<b>DESVIO PADRÃO</b>	1.4051
<b>ASSIMETRIA</b>	-0.01
<b>CURTOSE</b>	-1.21

FONTE: AUTOR (2018).

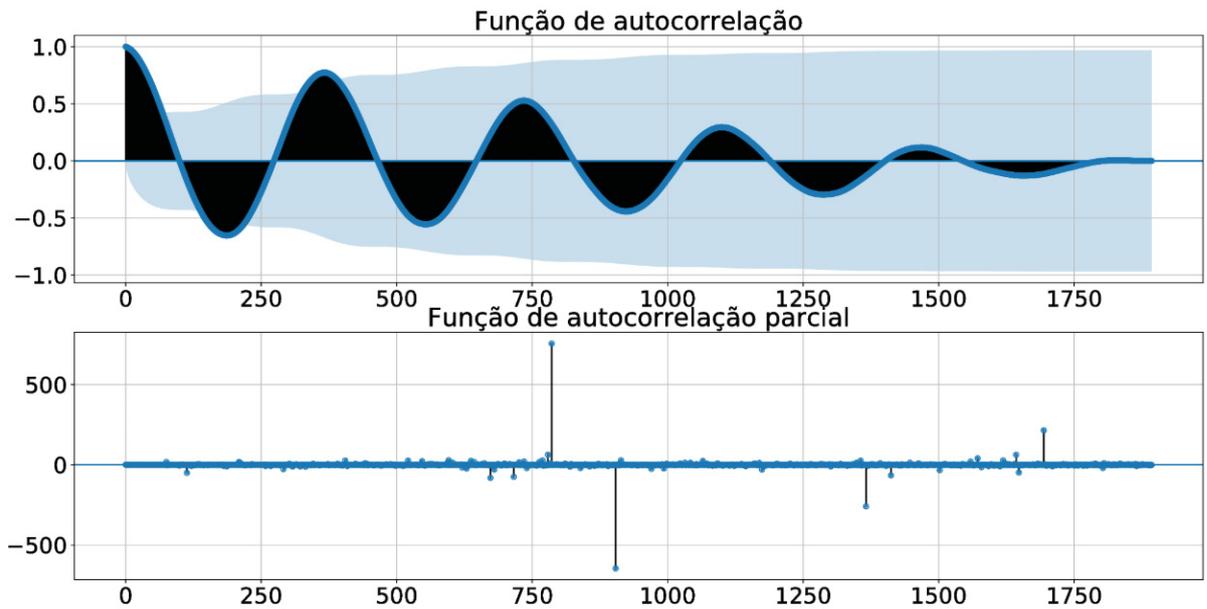
Na TABELA 1 é possível observar os dados descritivos da série, o valor de curtose de -1,21 indica que a série tem uma distribuição mais plana que uma distribuição normal com o mesmo valor da média e desvio padrão, também pode-se observar que apresenta uma assimetria bastante baixa devido a periodicidade observada na autocorrelação total.

GRÁFICO 1 SÉRIE TEMPORAL F05X



FONTE: AUTOR (2018).

GRÁFICO 2 FAC E FACP DA SÉRIE F05X



FONTE: AUTOR (2018).

## 4.2 SÉRIE TEMPORAL DO DESLOCAMENTO DO BLOCO F05 NO EIXO Y

No GRÁFICO 3 também é possível observar os dados de treinamento, validação e teste e no GRÁFICO 4 os gráficos da função de autocorrelação total e parcial. A série temporal apresenta autocorrelação total do tipo senóide amortecido como no eixo X deste bloco. Ademais apresenta vários picos na autocorrelação parcial com uma certa semelhança a uma oscilação decrescente mais ainda assim é possível notar o corte brusco. Igualmente à série temporal anterior, esta série pode ser modelada linearmente devido a exibição de picos fora do intervalo de confiança tanto na FAC como na FACP apresentando uma alta componente sistemática.

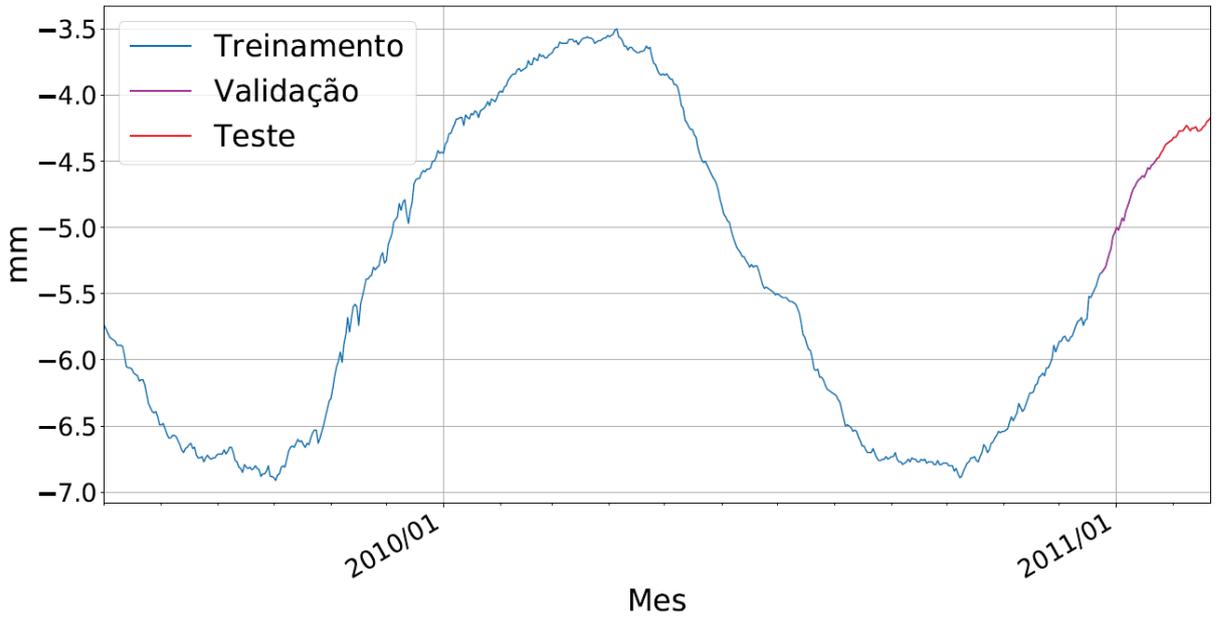
TABELA 2 DADOS DESCRITIVOS DA SÉRIE TEMPORAL F05Y

MEDIDA	RESULTADO
<b>MEDIA</b>	-5.0597571
<b>MIN</b>	-6.91
<b>MAX</b>	-3.3
<b>DESVIO PADRÃO</b>	1.1193
<b>ASSIMETRIA</b>	-0.089
<b>CURTOSE</b>	-1.43

FONTE: AUTOR (2018).

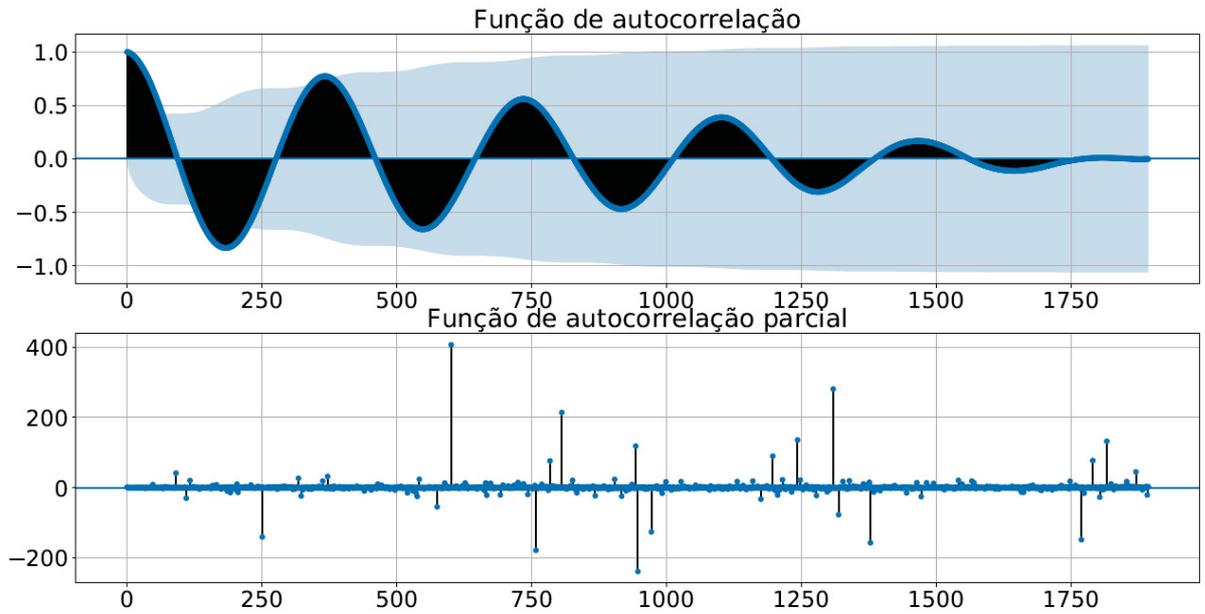
Na TABELA 2 é possível observar os dados descritivos da série, o valor de curtose de -1.43 e a assimetria -0.089, ambos continuam na mesma linha que a do eixo X. Neste caso o intervalo de variação foi mais pequeno observando os valores máximo e mínimo da série e o desvio padrão do mesmo. Vendo ambos deslocamentos é possível intuir que ambos têm um comportamento similar e correlacionado.

GRÁFICO 3 SÉRIE TEMPORAL F05Y



FONTE: AUTOR (2018).

GRÁFICO 4 FAC E FACP DA SÉRIE F05Y



FONTE: AUTOR (2018).

### 4.3 SÉRIE TEMPORAL DO DESLOCAMENTO DO BLOCO F13 NO EIXO X

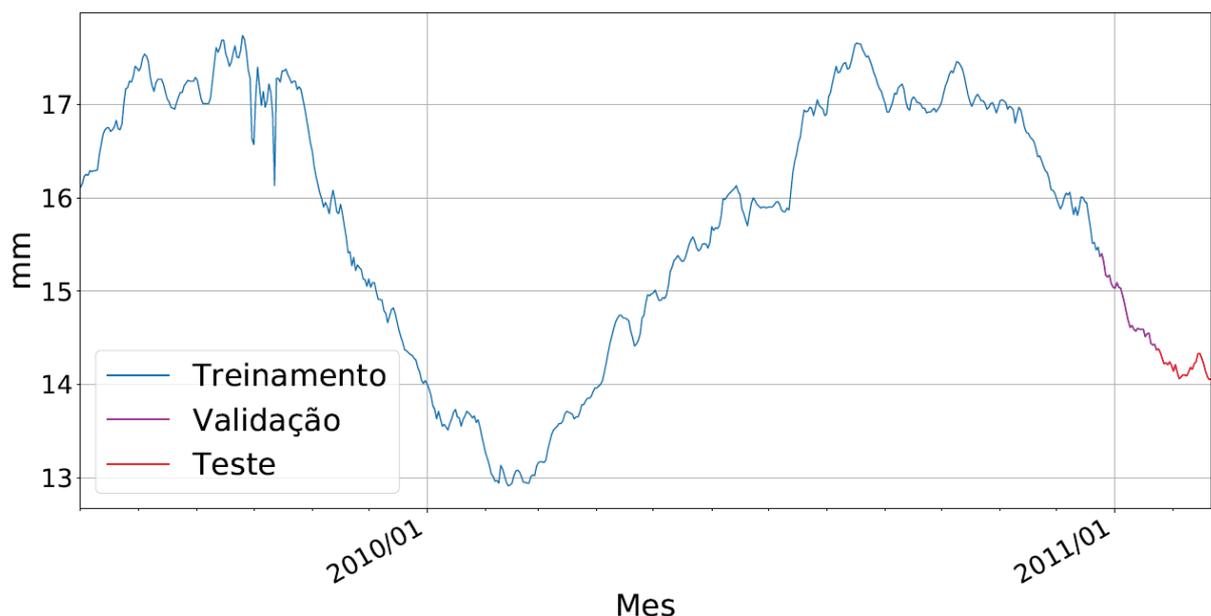
Assim como nos anteriores no GRÁFICO 5 e o GRÁFICO 6 é possível observar o gráfico da série temporal do bloco e o gráfico de sua FAC e FACP. Apresenta o mesmo comportamento que as demais séries temporais, uma senóide amortecida na FAC e picos na FACP, tendo isto podemos dizer que existe uma parte sistemática que pode ser modelada com um modelo ARIMA ou similar. Na TABELA 3 podem se ver a análise descritiva da série. O resultado indica que comparte as mesmas características que as anteriores.

TABELA 3 DADOS DESCRITIVOS DA SÉRIE TEMPORAL F13X

MEDIDA	RESULTADO
<b>MEDIA</b>	14.9
<b>MIN</b>	-3.75
<b>MAX</b>	-1.81
<b>DESVIO PADRÃO</b>	1.482
<b>ASSIMETRIA</b>	0.032
<b>CURTOSE</b>	-1.17

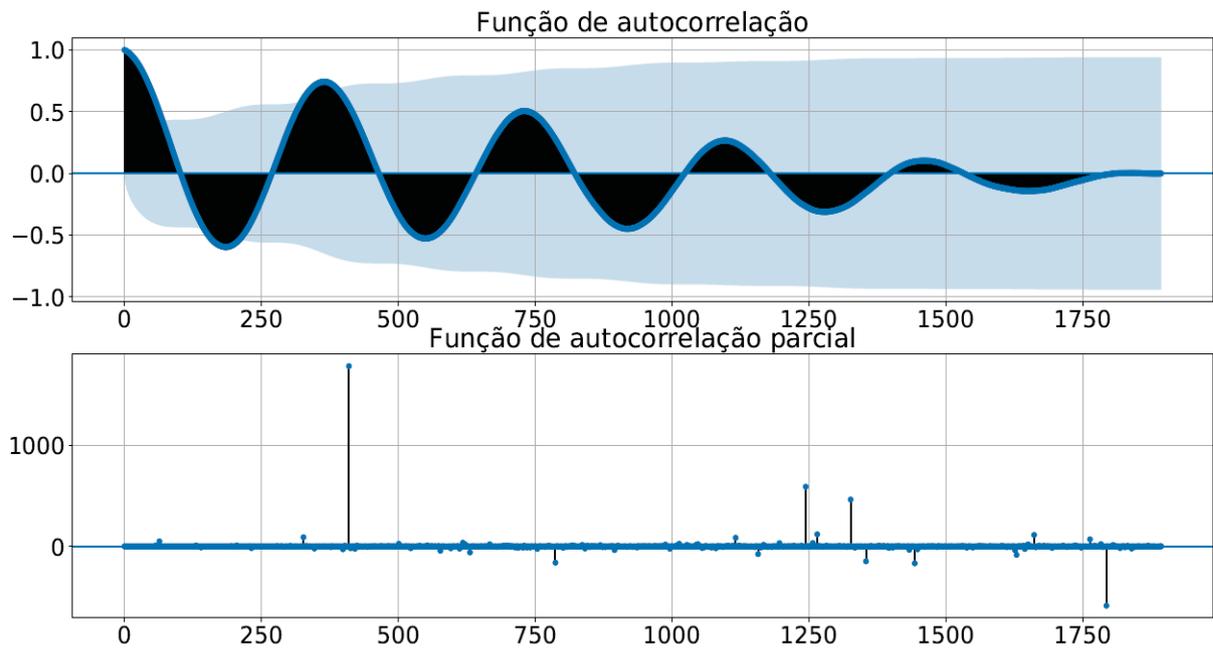
FONTE: AUTOR (2018).

GRÁFICO 5 SÉRIE TEMPORAL F13X



FONTE: AUTOR (2018).

GRÁFICO 6 FAC E FACP DA SÉRIE F13X



FONTE: AUTOR (2018).

#### 4.4 SÉRIE TEMPORAL DO DESLOCAMENTO DO BLOCO F13 NO EIXO Y

No GRÁFICO 7 e no GRÁFICO 8 pode-se observar os gráficos, tanto da série como da FAC e FACP. Os gráficos mostram as mesmas características que nas séries temporais anteriores, uma senóide amortecida na FAC e picos na FACP com a diferença de que os dados têm os picos de maior amplitude no lado negativo da curva. Também um modelo ARIMA poderia captar a parte sistemática do sistema.

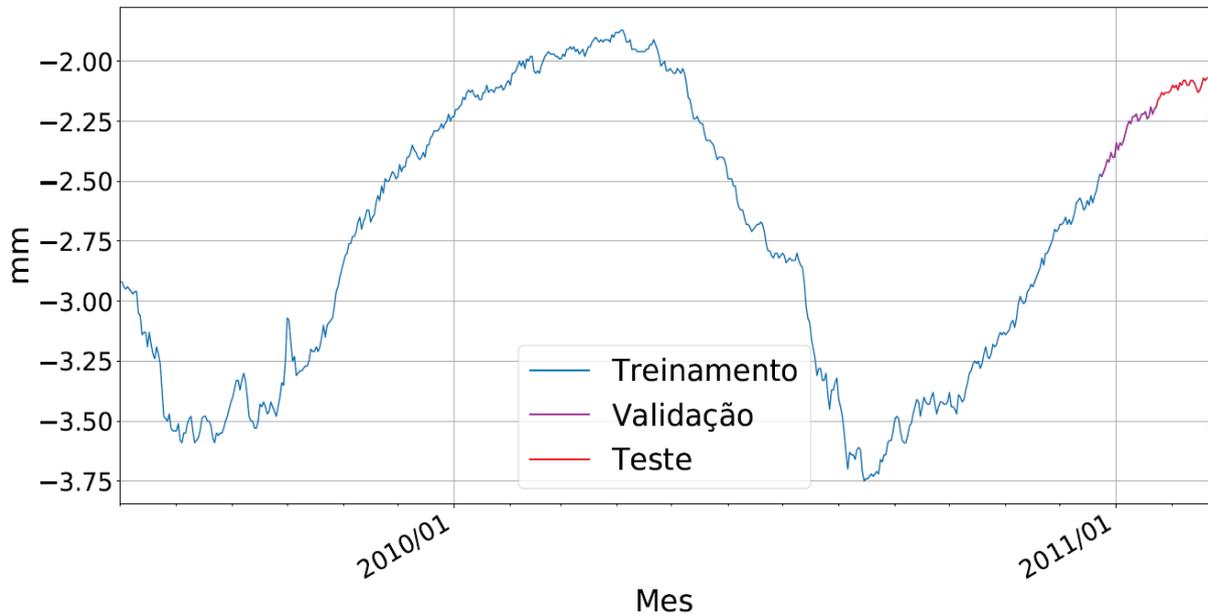
TABELA 4 DADOS DESCRITIVOS DA SÉRIE TEMPORAL F13Y

MEDIDA	RESULTADO
<b>MÉDIA</b>	-2.5987
<b>MIN</b>	-3.75
<b>MAX</b>	-1.81
<b>DESVIO PADRÃO</b>	0.5273
<b>ASSIMETRIA</b>	-0.4036954
<b>CURTOSE</b>	-1.10

FONTE: AUTOR (2018).

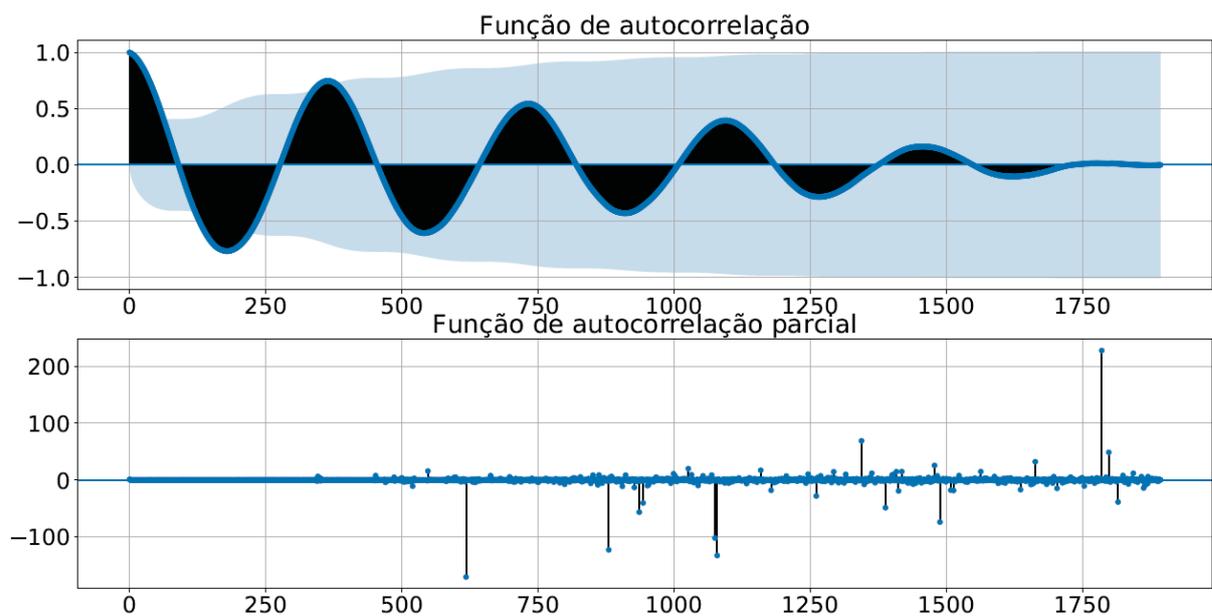
Na TABELA 4 está a análise descritiva, sendo a mesma que as anteriores com a diferença de que apresenta uma maior assimetria.

GRÁFICO 7 SÉRIE TEMPORAL F13Y



FONTE: AUTOR (2018).

GRÁFICO 8 FAC E FACP DA SÉRIE F13Y



FONTE: AUTOR (2018).

#### 4.5 SÉRIE TEMPORAL DO DESLOCAMENTO DO BLOCO F19 NO EIXO X

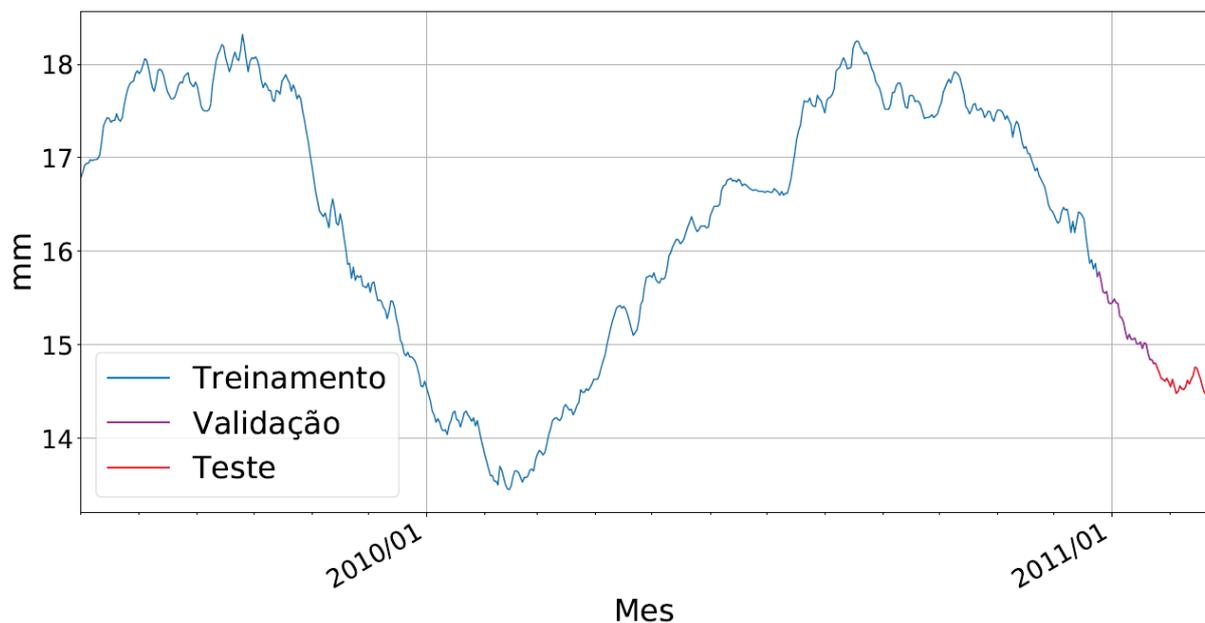
Observando o GRÁFICO 9 e o GRÁFICO 10, podemos notar os mesmos padrões que nas séries temporais anteriores e a mesmo ocorre observando a TABELA 5 com a análise descritiva.

TABELA 5 DADOS DESCRITIVOS DA SÉRIE TEMPORAL F19X

MEDIDA	RESULTADO
<b>MÉDIA</b>	15.52
<b>MIN</b>	12.87
<b>MAX</b>	18.32
<b>DESVIO PADRÃO</b>	1.479
<b>ASSIMETRIA</b>	-0.05
<b>CURTOSE</b>	-1.20

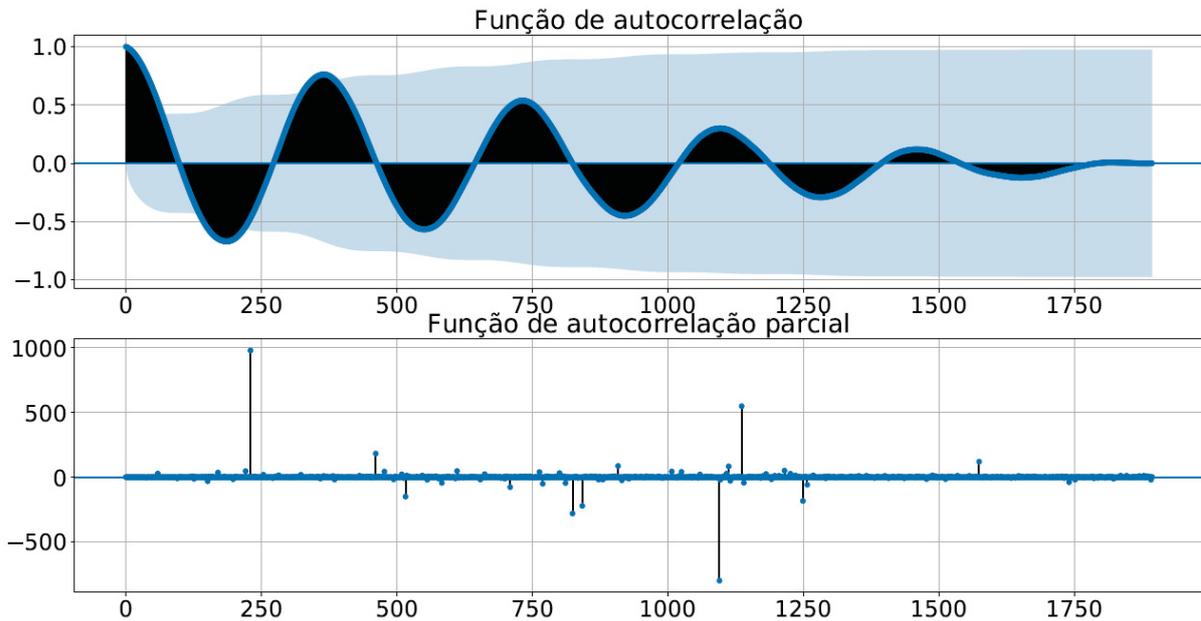
FONTE: AUTOR (2018).

GRÁFICO 9 SÉRIE TEMPORAL F19X



FONTE: AUTOR (2018).

GRÁFICO 10 FAC E FACP DA SÉRIE F19X



FONTE: AUTOR (2018).

#### 4.6 SÉRIE TEMPORAL DO DESLOCAMENTO DO BLOCO F19 NO EIXO Y

Com diferença das outras séries, no GRÁFICO 11 é possível notar muito mais ruído e menos periodicidade, com um comportamento mais estável a partir de janeiro de 2010 até maio do mesmo ano, onde bruscamente volta ao comportamento oscilante. Este comportamento é o mesmo para todo o ciclo de esta série temporal apresentando um comportamento menos oscilatório que os anteriores. O GRÁFICO 12 da FAC e FACP refletem esse comportamento, mas ainda assim apresentam a forma de senóide amortecido no primeiro e picos no segundo. Confirmando que uma análise com um modelo ARIMA seria possível.

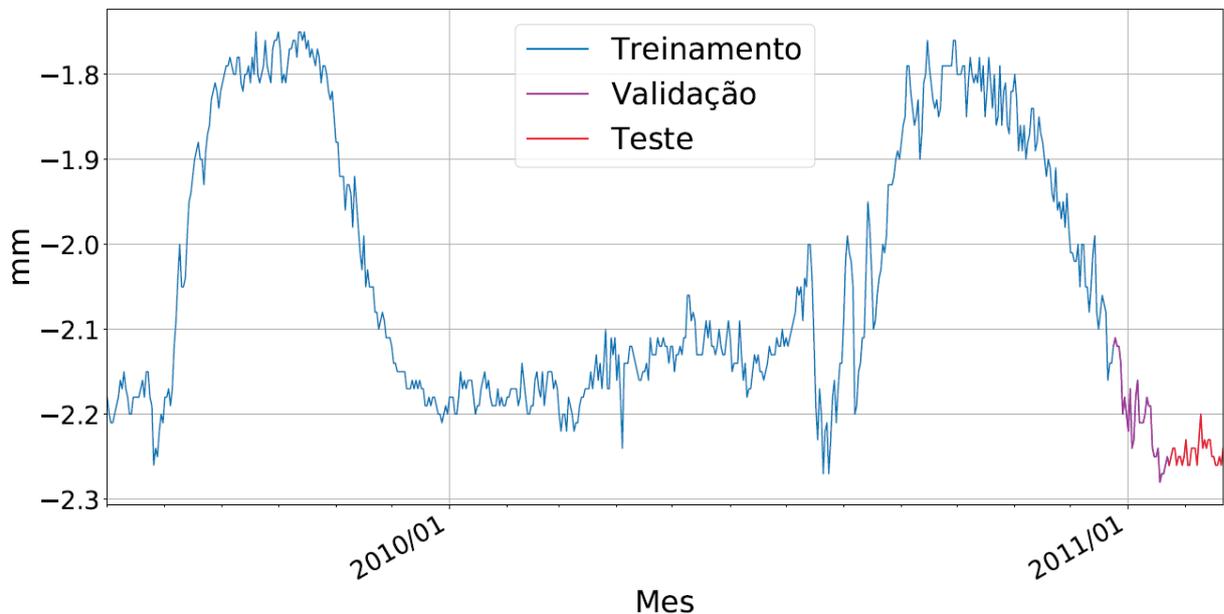
TABELA 6 DADOS DESCRITIVOS DA SÉRIE TEMPORAL F19Y

MEDIDA	RESULTADO
<b>MEDIA</b>	-2.0906
<b>MIN</b>	-2.34
<b>MAX</b>	-1.72
<b>DESVIO PADRÃO</b>	0.1535
<b>ASSIMETRIA</b>	0.906
<b>CURTOSE</b>	-0.551

FONTE: AUTOR (2018).

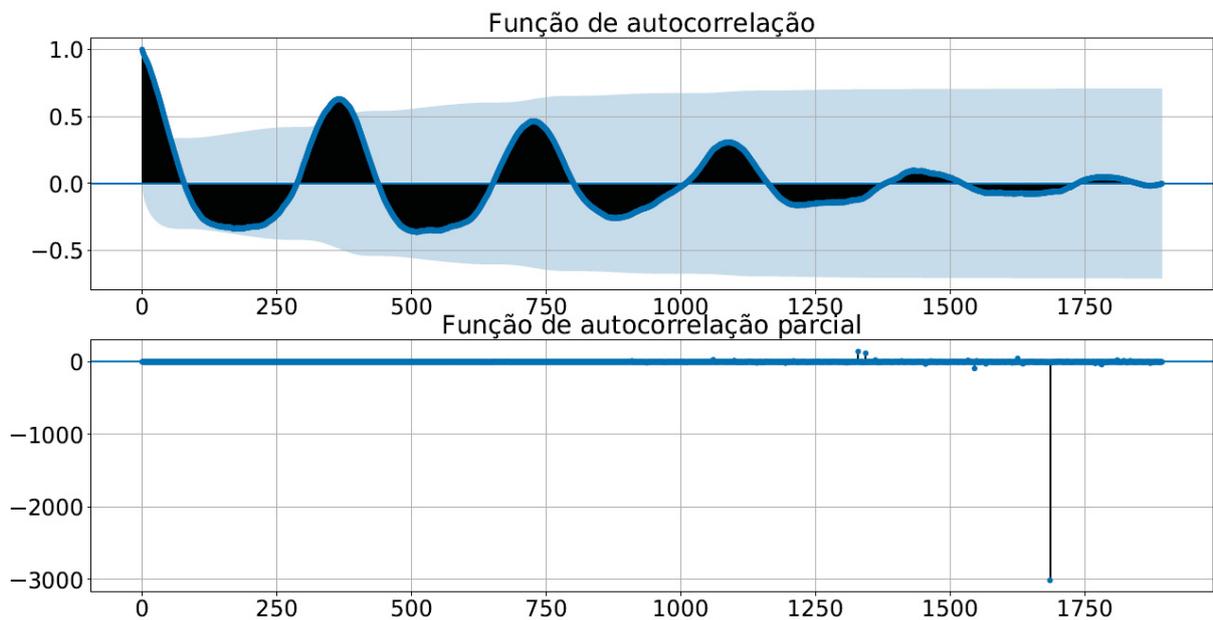
Na análise descritiva mostrada na TABELA 6 é possível notar analiticamente que existe muito mais assimetria nesta série e também que a curtose é mais próxima da distribuição normal sendo menos plana que as anteriores. Isto também é refletido nos valores de intervalo e desvio padrão que por consequência são menores, poderia ser que esses valores estabilizados se devam a injeção de consolidação nesse bloco (SILVEIRA; PEREIRA; FERREIRA, 2005).

GRÁFICO 11 SÉRIE TEMPORAL F19Y



FONTE: AUTOR (2018).

GRÁFICO 12 FAC E FACP DA SÉRIE F19Y



FONTE: AUTOR (2018).

#### 4.7 CONCLUSÃO DA ANÁLISE

Analisando todas as séries temporais, foi possível observar que todas, à exceção do deslocamento no eixo Y da série F19, tem um comportamento muito parecido uma com a outra, apresentando alta parte sistemática e tendo uma forma oscilatória no decorrer do tempo. Também avaliando a análise descritiva que foi feito em todas se pode observar que a distribuição dos dados tem uma simetria alta e tem uma forma mais plana que a distribuição gaussiana.

## 5 RESULTADOS

### 5.1 MODELAGEM DA SÉRIE DO DESLOCAMENTO DO BLOCO F05 NO EIXO X

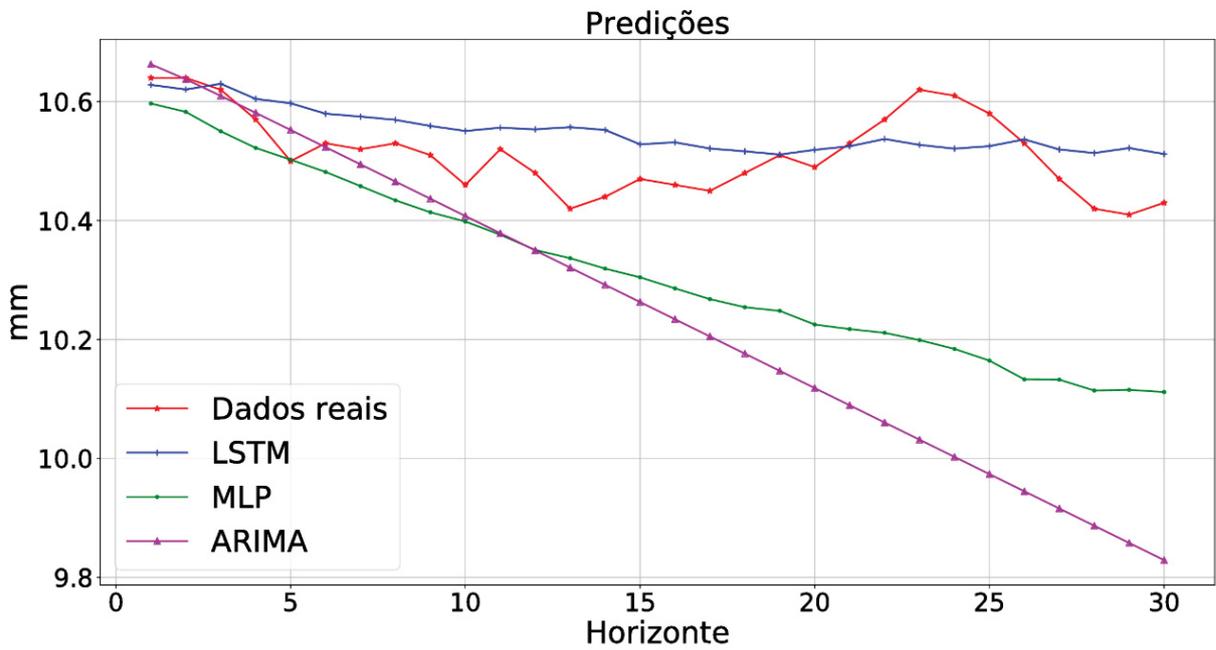
Na TABELA 7 estão os resultados das previsões dos melhores modelos encontrados, notavelmente o modelo neural com recorrência foi capaz de captar melhor a série com relação aos outros modelos. O erro MAPE foi menor a 1% com relação ao valor real. As previsões são mostradas no GRÁFICO 13 e a progressão do erro pode ser vista no GRÁFICO 14, mostrando que a rede LSTM mantém o nível da previsão em todo o horizonte. Para finalizar, foi feita a análise dos resíduos do melhor modelo com um nível de confiança de 95 e 99% mostrado no GRÁFICO 15 mostrando assim que praticamente toda a parte sistemática foi captada no modelo na previsão.

TABELA 7 RESULTADOS DO MODELADO DA SÉRIE F05X

Tipo	Validação			Teste			Configuração
	MAE	MAPE	MSE	MAE	MAPE	MSE	
<b>LSTM</b>	<b>0.0439</b>	<b>0.39</b>	<b>0.0027</b>	<b>0.0566</b>	<b>0.54</b>	<b>0.0044</b>	<b>E:10 N:20 S:30</b>
MLP	0.05144	0.4674	0.0037	0.1973	1.877	0.0563	E:10 N:10 S:30
ARIMA	0.11383	1.038	0.017	0.271	2.58	0.1245	AR:2 I:2 MA:2

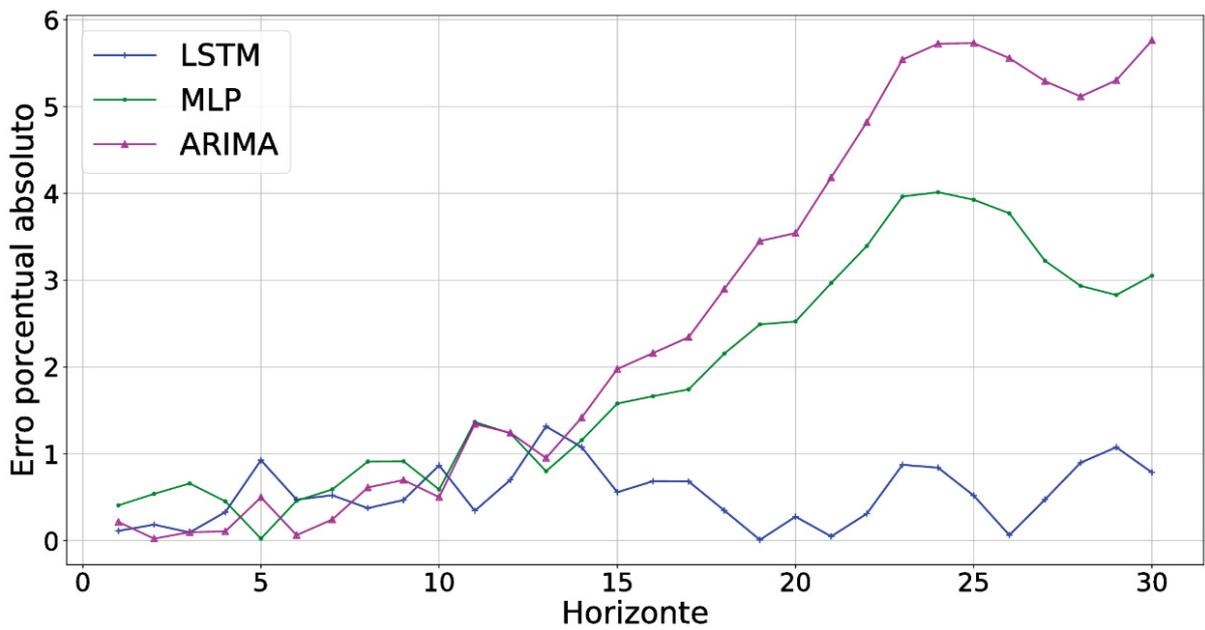
FONTE: AUTOR (2018).

GRÁFICO 13 PREVISÕES DA SÉRIE F05X



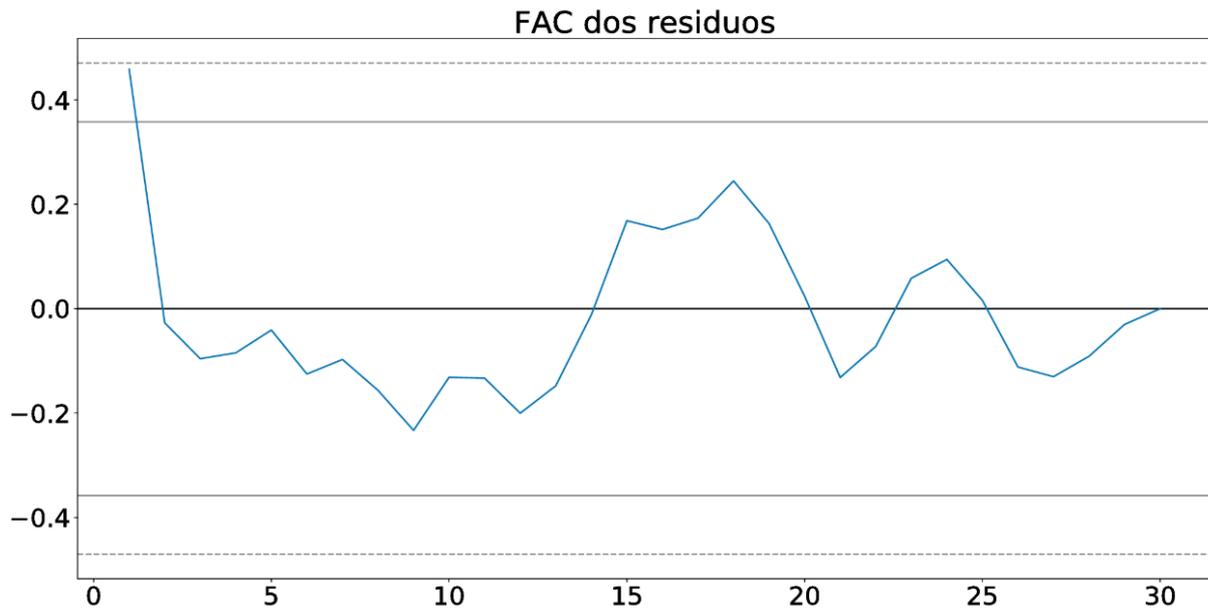
FONTE: AUTOR (2018).

GRÁFICO 14 PROGRESSÃO DO ERRO PORCENTUAL ABSOLUTO NA F05X



FONTE: AUTOR (2018).

GRÁFICO 15 FAC DOS RESÍDUOS DAS PREVISÕES DA SÉRIE F05X



FONTE: AUTOR (2018).

## 5.2 MODELAGEM DA SÉRIE DO DESLOCAMENTO DO BLOCO F05 NO EIXO Y

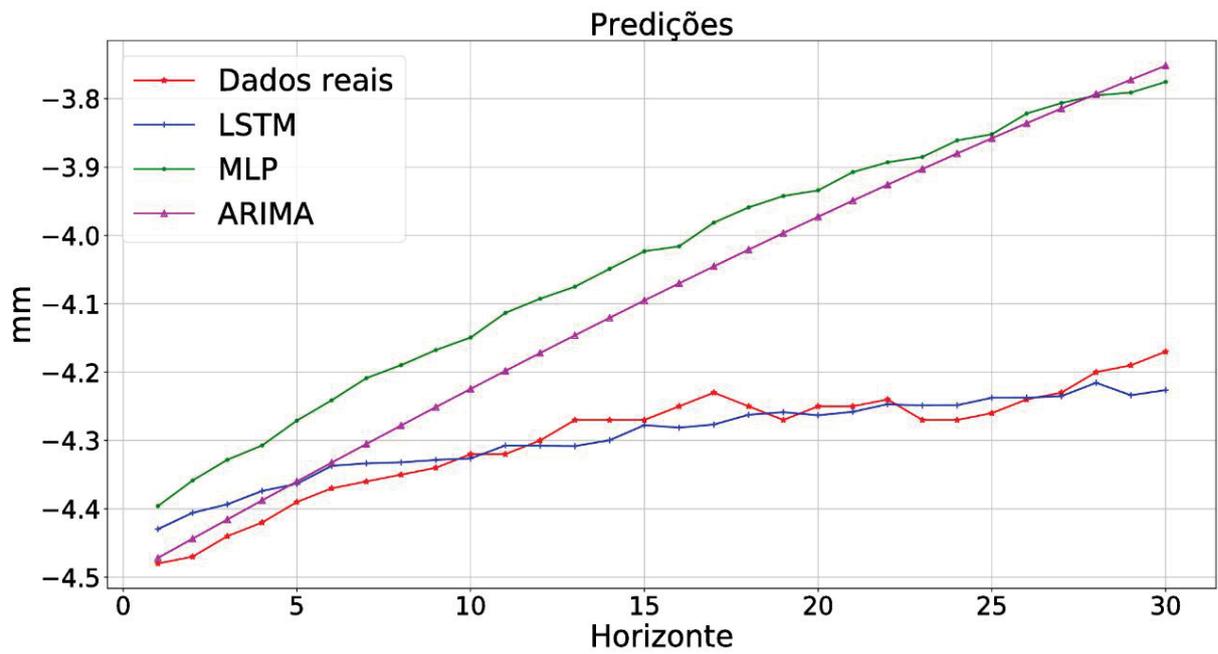
Na TABELA 8 são exibidos os resultados dos melhores modelos para a série F05Y, em que novamente o modelo da estrutura proposta foi melhor que os demais. Ou seja, o resultado foi mais notável pois o LSTM manteve a margem de erro por abaixo dos 2% perante os outros que tiveram um erro médio de 5 a 6%. Tudo isso pode ser visualizado no GRÁFICO 16 e no GRÁFICO 17. Também foi feita a FAC dos resíduos da previsão no GRÁFICO 18 verificando-se que a parte sistemática foi captada nas previsões.

TABELA 8 RESULTADOS DO MODELADO DA SÉRIE F05Y

Tipo	Validação			Teste			Configuração
	MAE	MAPE	MSE	MAE	MAPE	MSE	
LSTM	0.0323	0.65	0.0017	0.0247	0.57	0.009	E:10 N:10 S:30
MLP	0.0325	0.6683	0.0014	0.258	6.05	0.0792	E:10 N:10 S:30
ARIMA	0.0396	0.83	0.002	0.205	4.82	0.062	AR:2 I:0 MA:2

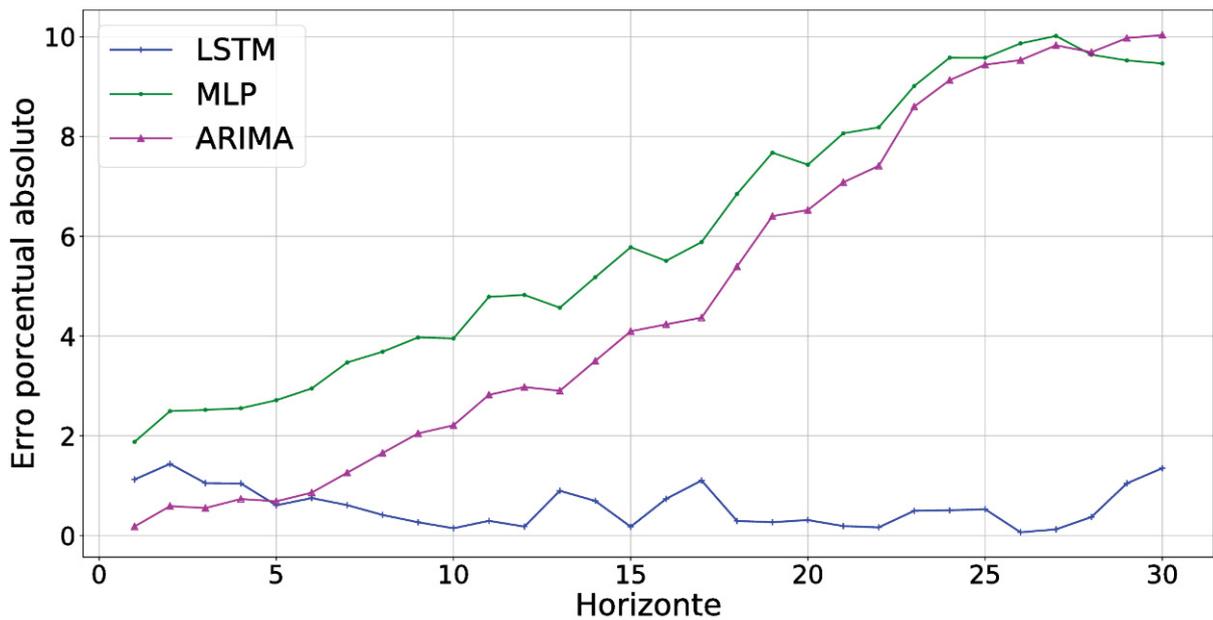
FONTE: AUTOR (2018).

GRÁFICO 16 DAS PREVISÕES DA SÉRIE F05Y



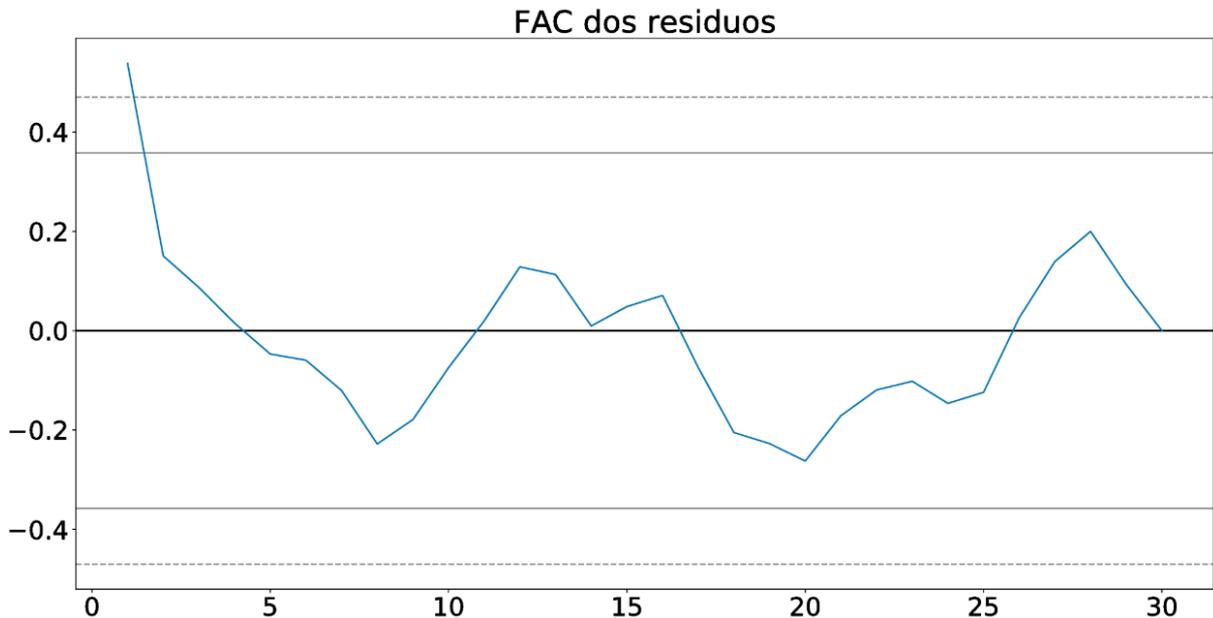
FONTE: AUTOR (2018).

GRÁFICO 17 PROGRESSÃO DO ERRO PORCENTUAL ABSOLUTO NA F05Y



FONTE: AUTOR (2018).

GRÁFICO 18 FAC DOS RESÍDUOS DAS PREVISÕES DA SÉRIE F05Y



FONTE: AUTOR (2018).

### 5.3 MODELAGEM DA SÉRIE DO DESLOCAMENTO DO BLOCO F13 NO EIXO X

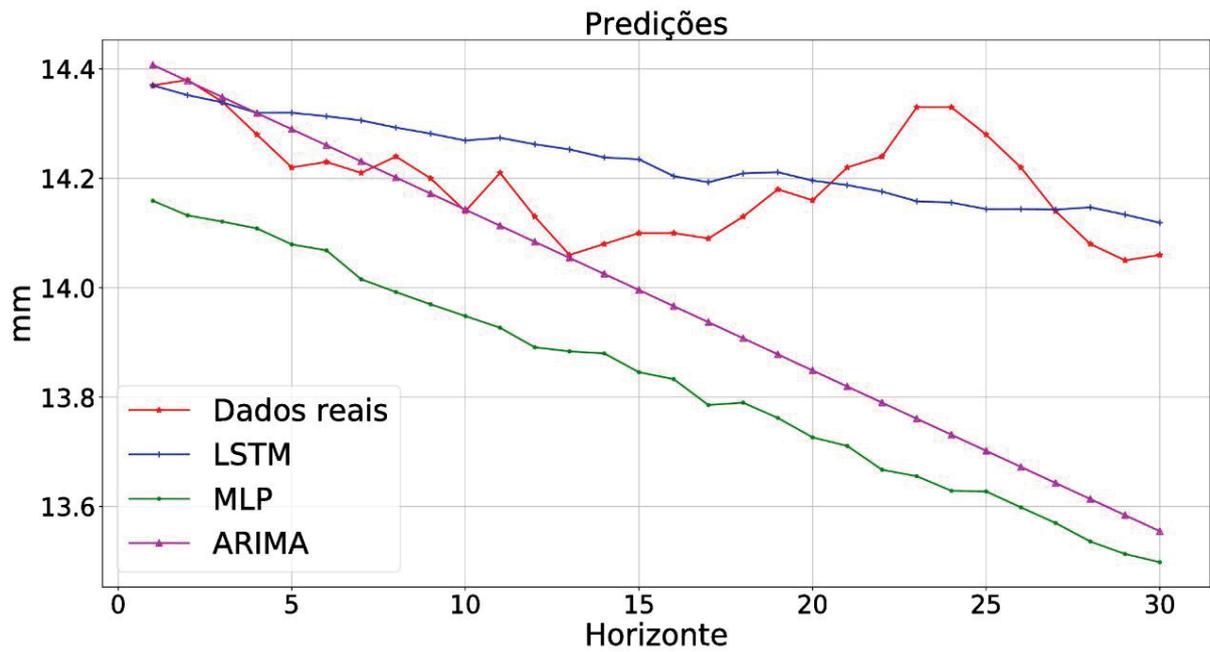
Assim como nos anteriores, na TABELA 9 pode se observar os resultados do modelado para a série F13X, uma vez mais a estrutura proposta foi a que melhor desempenho teve no teste, a recorrência do modelo permitiu ao mesmo detectar melhor a inclinação que estava fazendo a curva comparado aos outros modelos, isto pode ser visto no gráfico das previsões no GRÁFICO 19 e o modelo manteve o erro menor a 2% o que pode ser visualizado no GRÁFICO 20. A FAC dos resíduos da previsão mostrado no GRÁFICO 21 demonstra que o modelo captou a parte sistemática do mesmo na previsão.

TABELA 9 RESULTADOS DO MODELADO DA SÉRIE F13X

Tipo	Validação			Teste			Configuração
	MAE	MAPE	MSE	MAE	MAPE	MSE	
<b>LSTM</b>	<b>0.0569</b>	<b>0.384</b>	<b>0.0048</b>	<b>0.0838</b>	<b>0.591</b>	<b>0.0097</b>	<b>E:10 N:30 S:30</b>
MLP	0.0693	0.4672	0.0077	0.3625	2.5	0.1634	E:30 N:20 S:30
ARIMA	0.24765	1.6	0.0707	0.2262	1.59	0.0977	AR:0 I:2 MA:2

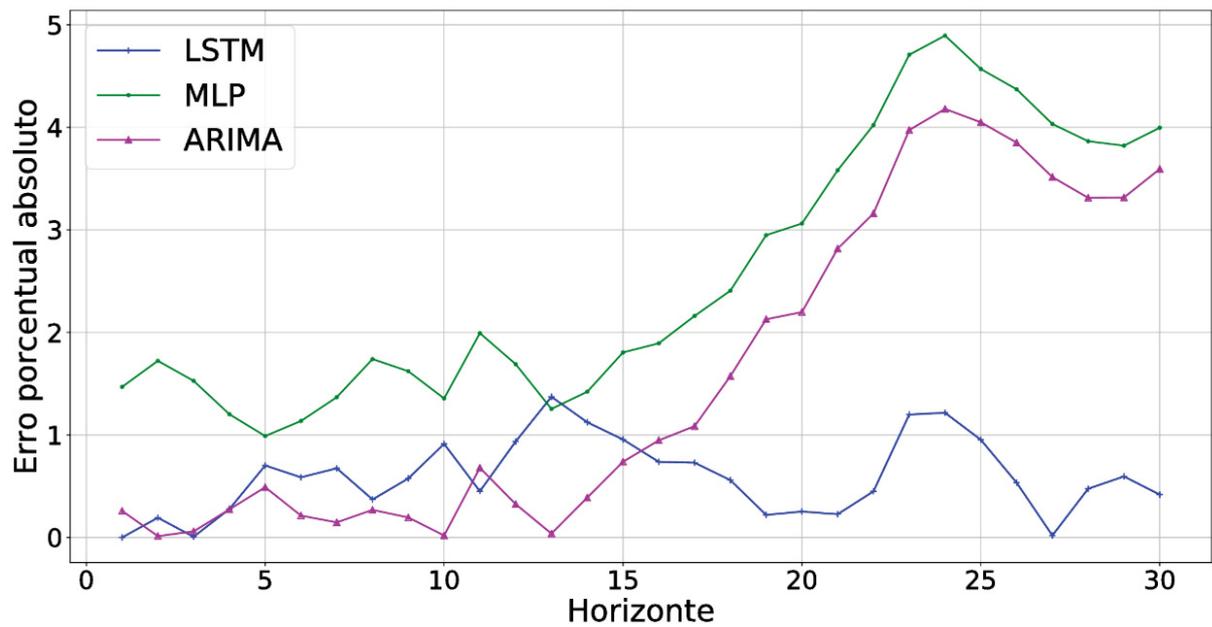
FONTE: AUTOR (2018).

GRÁFICO 19 PREVISÕES DA SÉRIE F13X



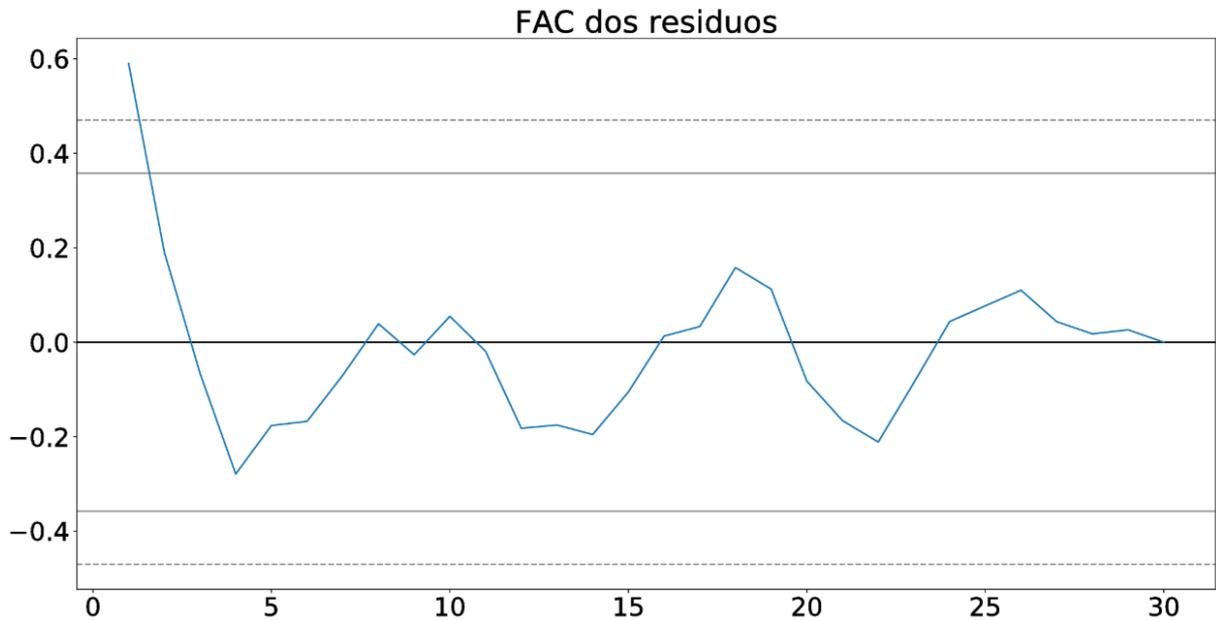
FONTE: AUTOR (2018).

GRÁFICO 20 PROGRESSÃO DO ERRO PORCENTUAL ABSOLUTO NA F13X



FONTE: AUTOR (2018).

GRÁFICO 21 FAC DOS RESÍDUOS DAS PREVISÕES DA SÉRIE F13X



FONTES: AUTOR (2018).

#### 5.4 MODELAGEM DA SÉRIE DO DESLOCAMENTO DO BLOCO F13 NO EIXO Y

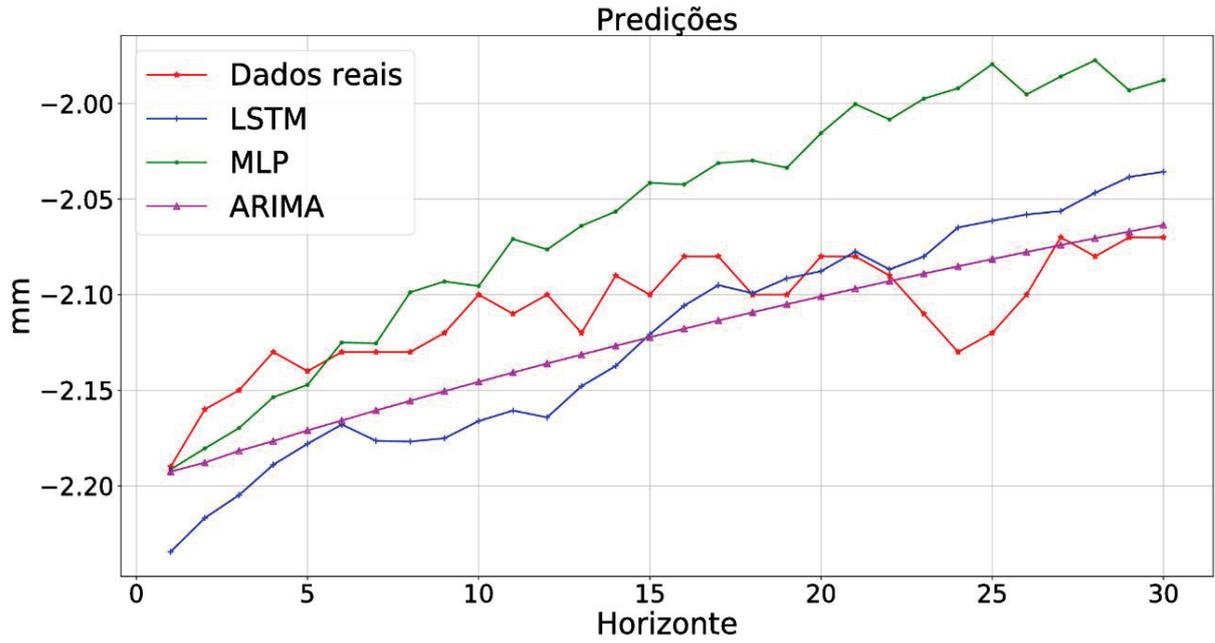
No TABELA 10 se pode ver o resultado dos modelos desta série, devido ao comportamento linear do segmento a ser predito o melhor modelo resultou ser o ARIMA, no é possível observar que a tendência é bastante linear e no GRÁFICO 22 e GRÁFICO 23 também pode se ver que a rede com LSTM ainda assim foi capaz de prever com um erro menor a 3% e estando bastante perto do modelo ARIMA. A FAC dos resíduos da previsão com o modelo ARIMA é mostrada no GRÁFICO 24, exibindo que a parte sistemática foi corretamente capturada.

TABELA 10 RESULTADOS DO MODELADO DA SÉRIE F13Y

Tipo	Validação			Teste			Configuração
	MAE	MAPE	MSE	MAE	MAPE	MSE	
LSTM	0.0174	0.76	4.6e-4	0.0363	1.71	0.0017	E:20 N:30 S:30
MLP	0.02130	0.932	7.6e-4	0.0548	2.612	0.0045	E:15 N:25 S:30
<b>ARIMA</b>	<b>0.04315</b>	<b>1.9</b>	<b>0.0026</b>	<b>0.0238</b>	<b>1.12</b>	<b>0.0007</b>	<b>AR:2 I:1 MA:2</b>

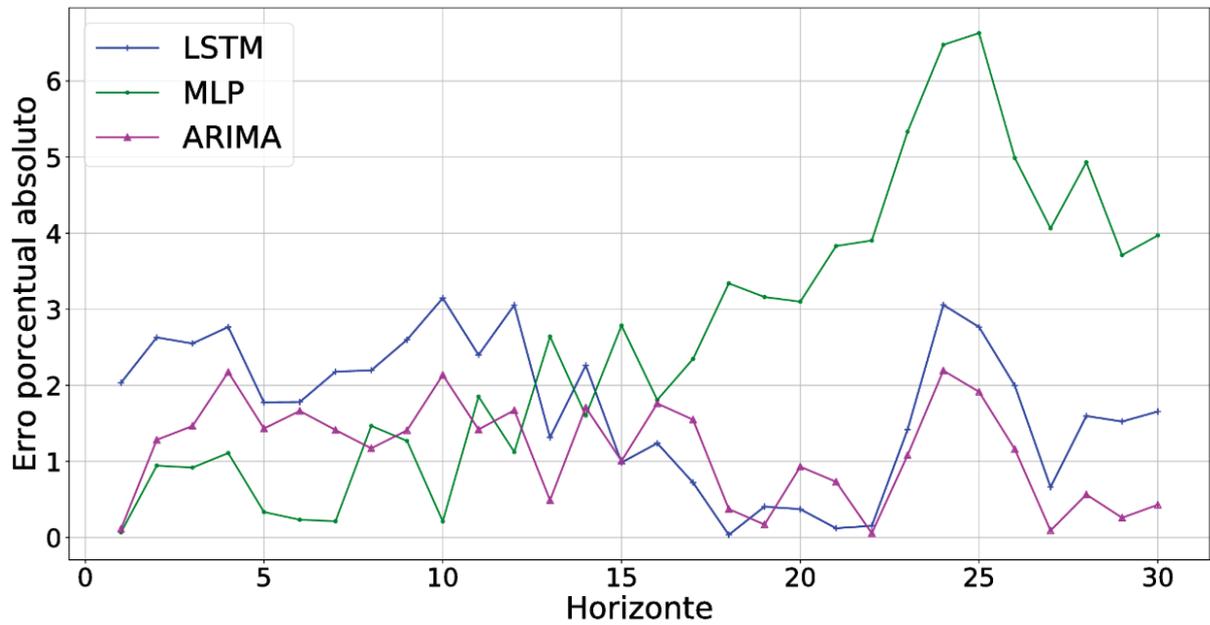
FONTES: AUTOR (2018).

GRÁFICO 22 PREVISÕES DA SÉRIE TEMPORAL F13Y



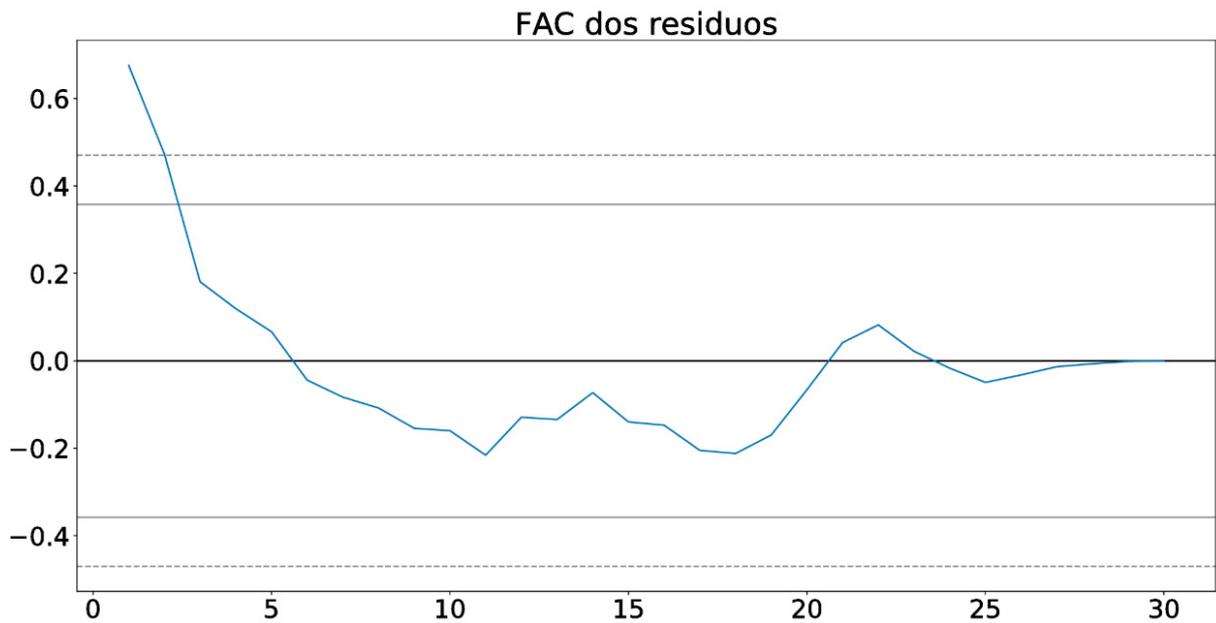
FONTE: AUTOR (2018).

GRÁFICO 23 PROGRESSÃO DO ERRO PORCENTUAL ABSOLUTO NA F13Y



FONTE: AUTOR (2018).

GRÁFICO 24 FAC DOS RESÍDUOS DAS PREVISÕES DA SÉRIE F13Y



FONTE: AUTOR (2018).

### 5.5 MODELADO DA SÉRIE DO DESLOCAMENTO DO BLOCO F19 NO EIXO X

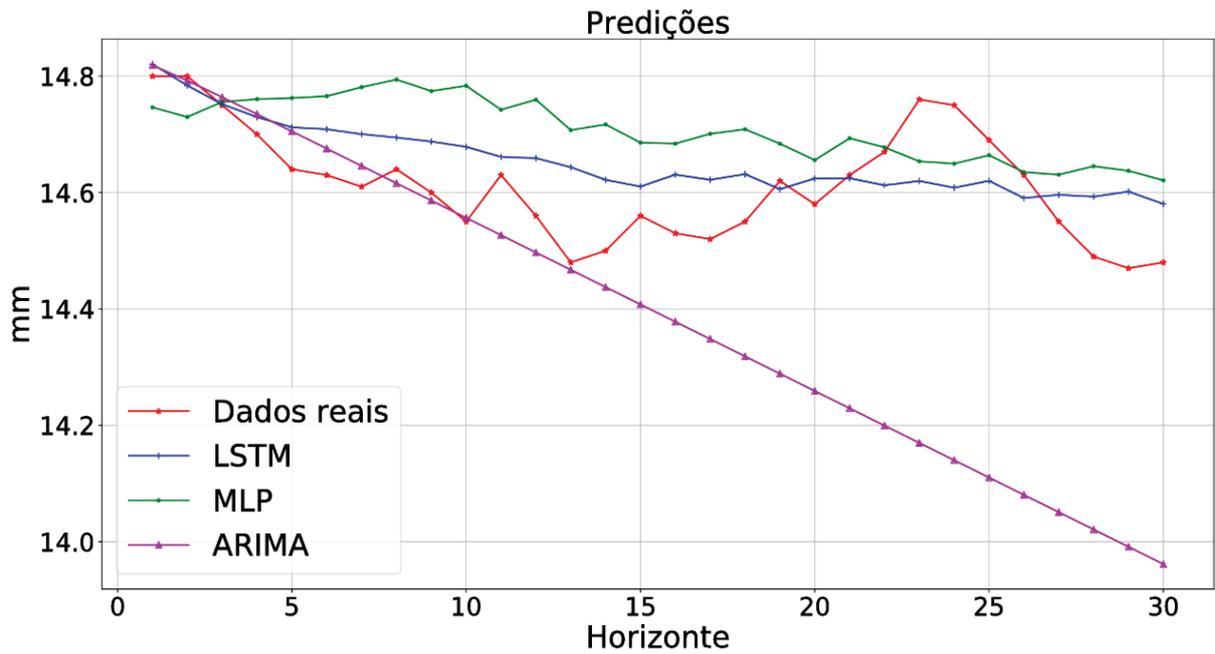
A TABELA 11 mostra que novamente o modelo com a rede LSTM foi o que melhor desempenho teve, no GRÁFICO 25 e no GRÁFICO 26 se pode ver a previsão e a progressão do erro que foi menor a 1.5% em todas as 30 previsões da rede do tipo LSTM. No GRÁFICO 27 a FAC dos resíduos das previsões é mostrada e notavelmente a parte sistemática foi capturada na previsão.

TABELA 11 RESULTADOS DO MODELADO DA SÉRIE F19X

Tipo	Validação			Teste			Configuração
	MAE	MAPE	MSE	MAE	MAPE	MSE	
<b>LSTM</b>	<b>0.0461</b>	<b>0.302</b>	<b>0.0029</b>	<b>0.0741</b>	<b>0.508</b>	<b>0.0074</b>	<b>E:10 N:10 S:30</b>
MLP	0.2265	1.495	0.0609	0.1183	0.8	0.0182	E:10 N:15 S:30
ARIMA	0.2566	1.69	0.0715	0.234	1.604	0.1026	AR:1 I:2 MA:2

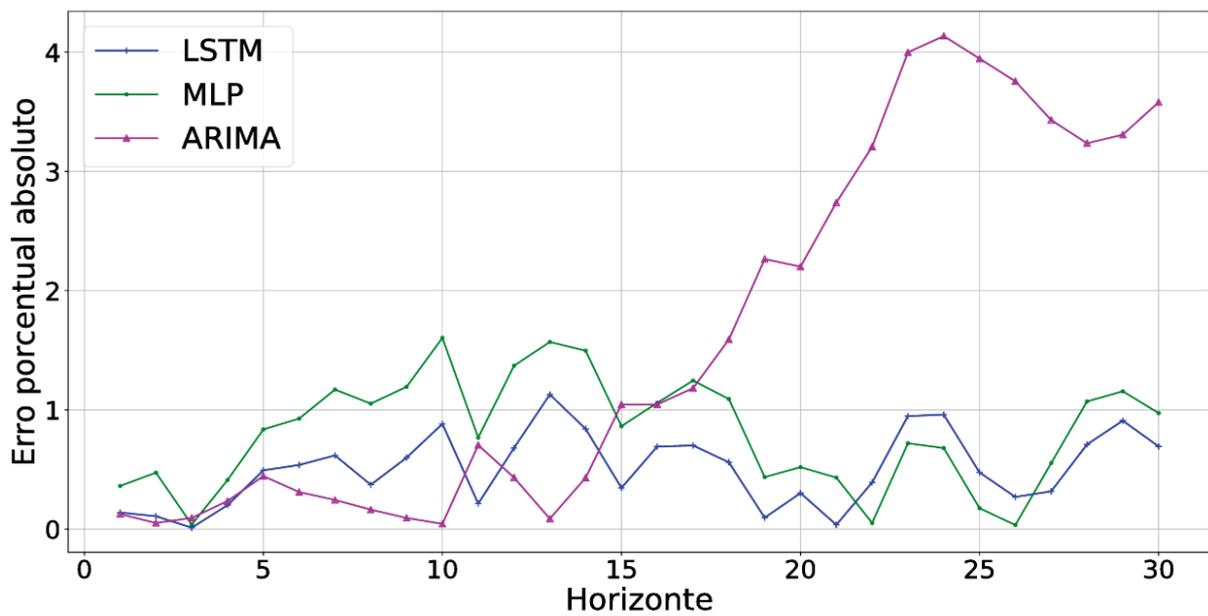
FONTE: AUTOR (2018).

GRÁFICO 25 PREVISÕES DA SÉRIE F19X



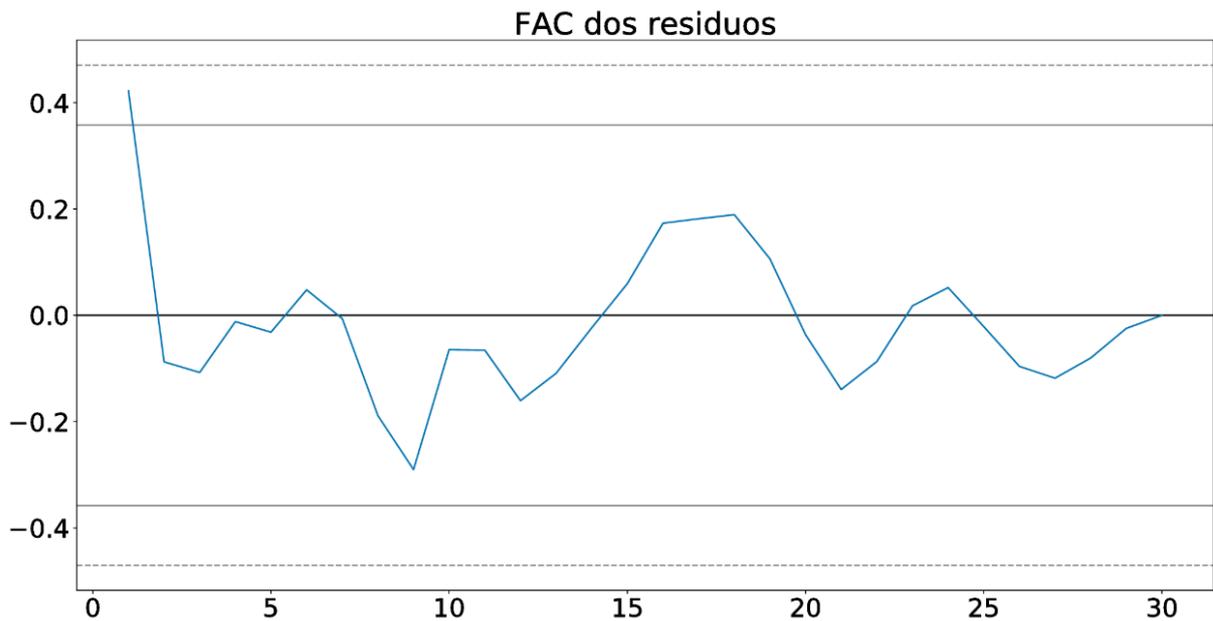
FONTE: AUTOR (2018).

GRÁFICO 26 PROGRESSÃO DO ERRO PORCENTUAL ABSOLUTO NA F19X



FONTE: AUTOR (2018).

GRÁFICO 27 FAC DOS RESÍDUOS DAS PREVISÕES DA SÉRIE F19X



FONTE: AUTOR (2018).

## 5.6 MODELADO DA SÉRIE DO DESLOCAMENTO DO BLOCO F19 NO EIXO Y

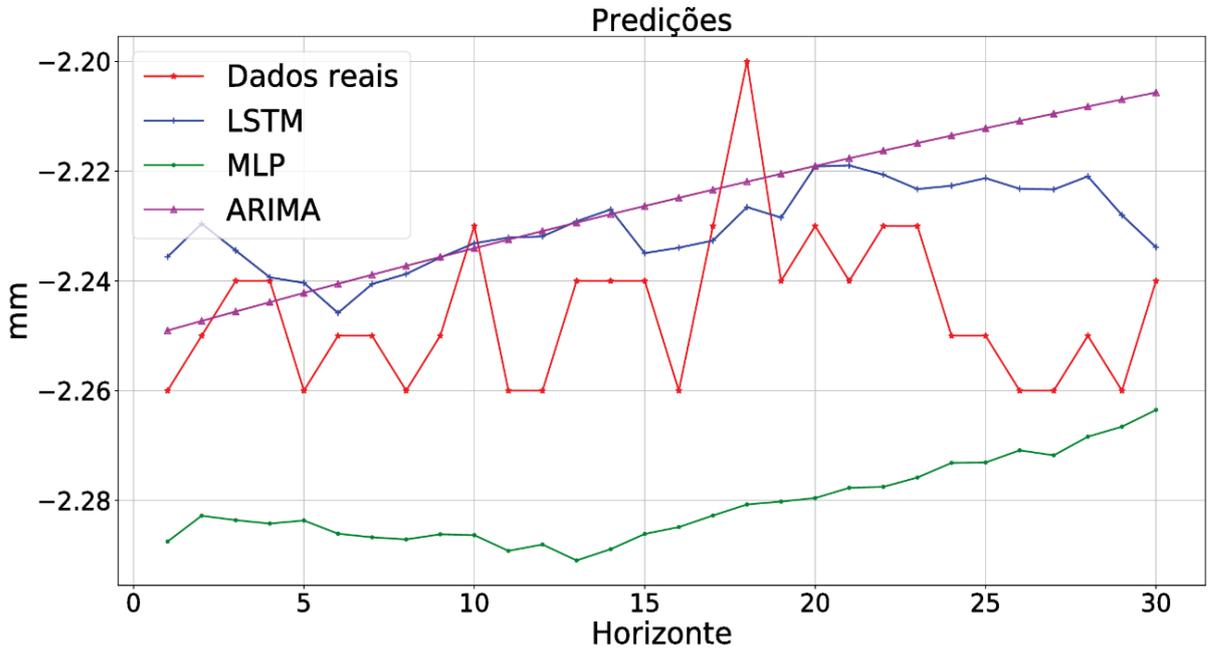
Nesta última série, novamente a rede do tipo LSTM foi a que melhor desempenho teve, na TABELA 12 é possível observar os dados dos erros e no GRÁFICO 28 o gráfico da previsão. Pode-se notar que o modelo ARIMA também capturou bem os dados e na progressão do erro estão muito próximos um do outro, conforme mostrado no GRÁFICO 29. A FAC no GRÁFICO 30 novamente mostra que nesta ocasião os resíduos não são diferenciáveis de um ruído aleatório, isto é, o modelo capturou completamente a parte sistemática da previsão.

TABELA 12 RESULTADOS DO MODELADO DA SÉRIE F19Y

Tipo	Validação			Teste			Configuração
	MAE	MAPE	MSE	MAE	MAPE	MSE	
LSTM	<b>0.02956</b>	<b>1.3</b>	<b>0.0013</b>	<b>0.0173</b>	<b>0.76</b>	<b>0.00041</b>	<b>E:25 N:5 S:30</b>
MLP	0.02911	1.3	0.0010	0.0355	1.5	0.0015	E:30 N:5 S:30
ARIMA	0.076	3.4	0.007	0.0214	0.9	0.00067	AR:1 I:0 MA:1

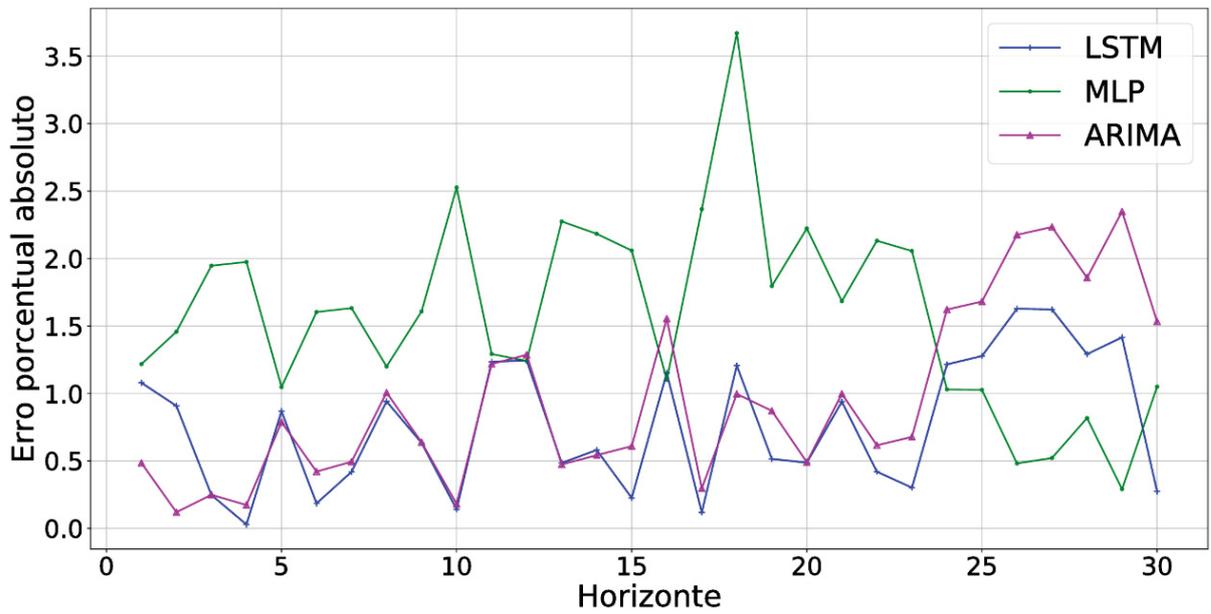
FONTE: AUTOR (2018).

GRÁFICO 28 PREVISÕES DA SÉRIE F19Y



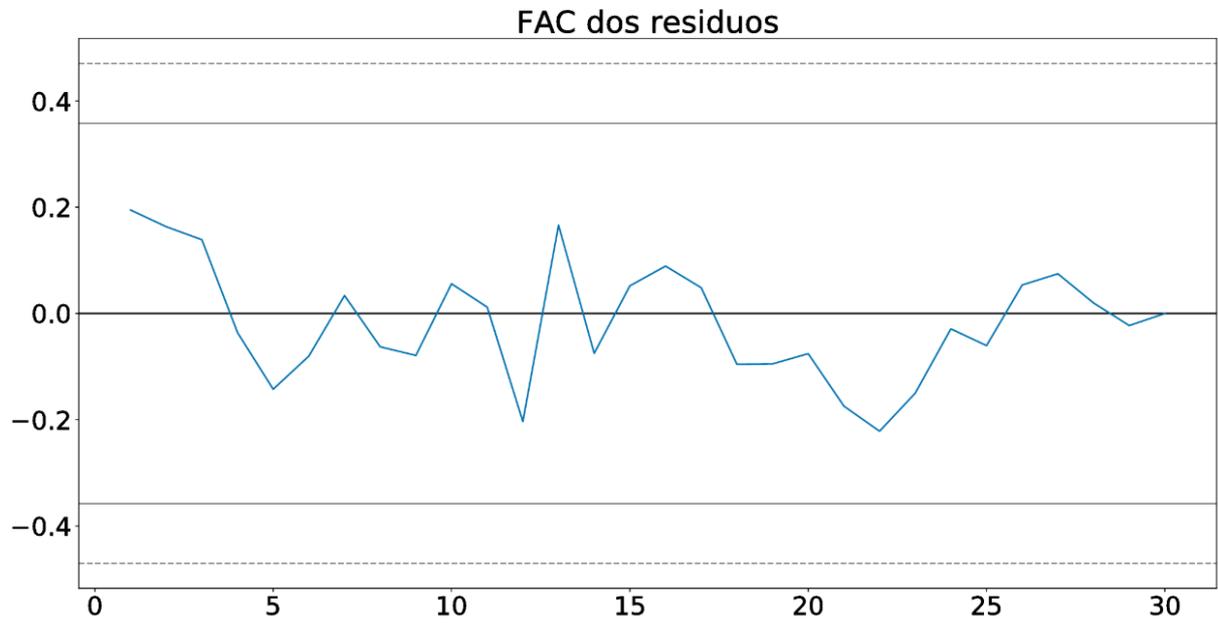
FONTE: AUTOR (2018).

GRÁFICO 29 PROGRESSÃO DO ERRO PORCENTUAL ABSOLUTO NA F19Y



FONTE: AUTOR (2018).

GRÁFICO 30 FAC DOS RESÍDUOS DAS PREVISÕES DA SÉRIE F19Y



FONTE: AUTOR (2018).

## 6 CONCLUSÕES

Após observar os resultados mostrados na seção anterior, pode-se concluir que:

- Os objetivos estabelecidos foram completamente alcançados.
- A análise foi feita nas séries temporais dando um panorama do comportamento estatístico das mesmas.
- Da comparação com os métodos tradicionais na maioria das séries temporais a estrutura utilizando LSTM teve melhor desempenho na previsão dos valores de teste da série.
- Foi implementado o método com a biblioteca Keras para a linguagem de programação Python cumprindo com o objetivo de utilizar uma biblioteca moderna para a modelagem das RNAs.
- Mais testes com outros tipos de séries temporais devem ser feitos para provar a robustez do método.

### 6.1 RECOMENDAÇÕES PARA TRABALHOS FUTUROS

- Utilizar uma decomposição temporal para trabalhar com as componentes da tendência e sazonalidade, em Armstrong (2007) se menciona que foram obtidos melhores previsões ao fazer-se esta decomposição.
- Utilizar outras técnicas de otimização de hiperparâmetros como a otimização bayesiana ou a busca aleatória (BUDUMA; LOCASCIO, 2017).
- Testar a estrutura num modelo híbrido, combinando a eficiência dos modelos tradicionais com a estrutura proposta aqui.
- Obter os intervalos de confiança das previsões do modelo utilizando séries sintéticas formadas com a técnica de amostragem *bootstrap* mostrada em Teixeira (2015).

## REFERÊNCIAS

- ABADI, M. et al. **Tensorflow: a system for large-scale machine learning**. OSDI. **Anais...** In: SYMPOSIUM ON OPERATING SYSTEMS DESIGN AND IMPLEMENTATION. 2016
- ARMSTRONG, J. S. **Principles of forecasting: a handbook for researchers and practitioners**. 7. print ed. New York: Springer, 2007.
- BANDARA, K.; BERGMEIR, C.; SMYL, S. Forecasting Across Time Series Databases using Long Short-Term Memory Networks on Groups of Similar Series. **arXiv preprint arXiv:1710.03222**, 2017.
- BIANCHI, M.; BREMEN, R. Health Monitoring of Arch Dams-Recent Developments. **Restoration of Buildings and Monuments**, v. 7, n. 3–4, p. 271–284, 2001.
- BINACIONAL, I. **Instrumentação**. Disponível em: <<https://www.itaipu.gov.br/energia/instrumentacao>>. Acesso em: 26 mar. 2018.
- BUDUMA, N.; LOCASCIO, N. **Fundamentals of deep learning: Designing next-generation machine intelligence algorithms**. First edition ed. Sebastopol, CA: O'Reilly Media, 2017.
- CHOLLET, F. **KERAS**. [s.l.] GitHub, 2015.
- CORRÊA, J. M. **Método WARIMAX-GARCH neural para previsão de séries temporais**. PhD Tesis—Curitiba: Universidade Federal do Paraná, 2015.
- CYBENKO, G. Approximation by superpositions of a sigmoidal function. **Mathematics of control, signals and systems**, v. 2, n. 4, p. 303–314, 1989.
- FUNAHASHI, K.-I. On the approximate realization of continuous mappings by neural networks. **Neural networks**, v. 2, n. 3, p. 183–192, 1989.
- GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep learning**. Cambridge, Massachusetts: The MIT Press, 2016.
- GRAHAM, W. J. **A procedure for estimating loss of life caused by dam failure**. Denver: Bureau of Reclamation, 1999.
- HAYKIN, S.; ENGEL, P. M. **Redes neurais: princípios e prática**. Porto Alegre: Bookman Editora, 2001.
- HAYKIN, S. S. **Neural networks and learning machines**. 3. ed. New York: Pearson, 2009.
- HERMANO. **Neurônio**. Disponível em: <[bit.ly/2PIBXcw](http://bit.ly/2PIBXcw)>. Acesso em: 10 set. 2018.
- HSIEH, T.-J.; HSIAO, H.-F.; YEH, W.-C. Forecasting stock markets using wavelet transforms and recurrent neural networks: An integrated system based on artificial bee colony algorithm. **Applied Soft Computing**, v. 11, n. 2, p. 2510–2525, mar. 2011.

KARPATHY, A. **The Unreasonable Effectiveness of Recurrent Neural Networks**. Disponível em: <<http://karpathy.github.io/2015/05/21/rnn-effectiveness/>>. Acesso em: 18 set. 2018.

KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. **arXiv preprint arXiv:1412.6980**, 2014.

KOKOSKA, S.; ZWILLINGER, D. **CRC standard probability and statistics tables and formulae**. Student ed ed. Boca Raton, Fla: Chapman & Hall/CRC, 2000.

KUBRUSLY, C. S. **Elements of operator theory**. Second edition ed. New York: Birkhäuser, 2011.

LAKSHAY, S. **Machine Learning: Supervised Vs Unsupervised Learning**. Blog. Disponível em: <[lakshaysuri.wordpress.com/2017/03/19/machine-learning-supervised-vs-unsupervised-learning/](http://lakshaysuri.wordpress.com/2017/03/19/machine-learning-supervised-vs-unsupervised-learning/)>. Acesso em: 22 nov. 2018.

LIEW, S. S.; KHALIL-HANI, M.; BAKHTERI, R. Bounded activation functions for enhanced training stability of deep neural networks on visual pattern recognition problems. **Neurocomputing**, v. 216, p. 718–734, dez. 2016.

LIU, H.; MI, X.; LI, Y. Wind speed forecasting method based on deep learning strategy using empirical wavelet transform, long short term memory neural network and Elman neural network. **Energy Conversion and Management**, v. 156, p. 498–514, jan. 2018.

MA, X. et al. Long short-term memory neural network for traffic speed prediction using remote microwave sensor data. **Transportation Research Part C: Emerging Technologies**, v. 54, p. 187–197, maio 2015.

MATA, J. Interpretation of concrete dam behaviour with artificial neural network and multiple linear regression models. **Engineering Structures**, v. 33, p. 903–910, 2011.

MITCHELL, T. M. **Machine Learning**. New York: McGraw-Hill, 1997.

P. BUKENYA, C. O., P. Moyo, H. Beushausen. Health monitoring of concrete dams: a literature review. **Journal of Civil Structural Health Monitoring**, v. 4, p. 235–244, 2014.

PALIT, A. K.; POPOVIC, D. **Computational Intelligence in Time Series Forecasting: Theory and Engineering Applications**. Dordrecht: Springer, 2005.

PARZEN, E. On spectral analysis with missing observations and amplitude modulation. **The Indian Journal of Statistics, Series A**, p. 383–392, 1963.

RICH, T. P. Lessons in social responsibility from the Austin dam failure. **International Journal of Engineering Education**, v. 22, p. 1287–1296, 2006.

SHI, H.; XU, M.; LI, R. Deep Learning for Household Load Forecasting—A Novel Pooling Deep RNN. **IEEE Transactions on Smart Grid**, v. 9, n. 5, p. 5271–5280, set. 2018.

SHI, X. et al. **Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting**. (C. Cortes et al., Eds.)Advances in Neural Information Processing Systems 28. **Anais...** In: CONFERENCE ON NEURAL INFORMATION PROCESSING SYSTEMS. Montreal, Canada: Curran Associates, Inc., 2015

SILVEIRA, J. F. A. **Revisão Crítica sobre a instrumentação de barragens no Brasil**. [s.l.] Comitê Brasileiro de Barragens, 2013.

SILVEIRA, J. F. A.; PEREIRA, P. N.; FERREIRA, W. V. F. Anomalia geológica na fundação das estruturas de concreto de canoas I e sua influência no comportamento da barragem. **COMITÊ BRASILEIRO DE BARRAGENS**, 2005.

SNYMAN, J. **Practical mathematical optimization: an introduction to basic optimization theory and classical and new gradient-based algorithms**. [s.l.] Springer Science & Business Media, 2005. v. 97

TEIXEIRA, L. L. **Projeção de séries temporais por meio de um método híbrido wavelet-neural integrado com bootstrap**. Curitiba: Universidade Federal do Paraná. Setor de Tecnologia. Programa de Pós-Graduação em Métodos Numéricos em Engenharia, 27 ago. 2015.

THOMAS, A. **Keras LSTM tutorial - How to easily build a powerful deep learning language model**. Disponível em: <<http://adventuresinmachinelearning.com/keras-lstm-tutorial/>>. Acesso em: 20 set. 2018.

WU, H. Global stability analysis of a general class of discontinuous neural networks with linear growth activation functions. **Information Sciences**, v. 179, n. 19, p. 3432–3441, 2009.

ZHANG, G. P. Neural Networks for Time-Series Forecasting. In: ROZENBERG, G.; BÄCK, T.; KOK, J. N. (Eds.). **Handbook of Natural Computing**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. p. 461–477.

## ANEXO 1 – ALGORITMOS UTILIZADOS NAS REDES NEURAIS

### Notação

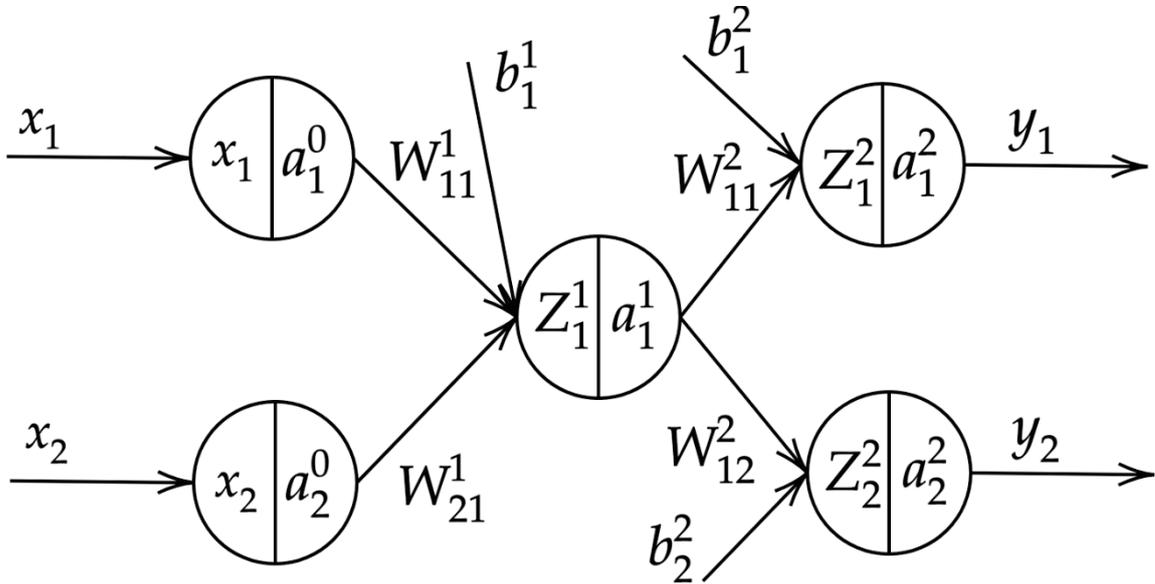
- $L$  Número total de camadas da rede
- $l$  Indexação da camada na forma  $l = 1, 2, 3, \dots, L$
- Neurônios da camada  $l$  serão indexados com  $j$
- Neurônios da camada  $l - 1$  serão indexados com  $k$
- $o_j$  valor desejado do neurônio  $j$  na última camada  $L$  para um dado de treinamento
- $Ct$  custo total da rede para um dado de treinamento
- $W_{kj}^l$  peso sináptico que conecta o neurônio  $k$  na camada  $l - 1$  com o neurônio  $j$  na camada  $l$
- $W_j^l$  vetor dos pesos sinápticos conectados ao neurônio  $j$  na camada  $l$  por cada um dos neurônios da camada  $l - 1$
- $W^l$  matriz dos pesos sináptico conectados aos neurônios da camada  $l$
- $Z_j^l$  a entrada do neurônio  $j$  na camada  $l$
- $f_l$  a função de ativação utilizada na camada  $l$
- $a_j^l$  a saída da função de ativação do neurônio  $j$  na camada  $l$
- $b_j^l$  o desvio da combinação linear no neurônio  $j$  na camada  $l$
- $x$  e  $y$  são a entradas e a saída da rede respectivamente

### Propagação para frente

Usando como exemplo do algoritmo de propagação para frente (*Forward Propagation*) uma rede de duas entradas, duas saídas e uma camada oculta com um só neurônio:

Sendo os vetores de entrada e saída dos respectivos neurônios os seguintes:

FIGURA 13 REDE NEURAL DE DUAS CAMADAS



$$\mathbf{a}^0 = \begin{bmatrix} a_1^0 \\ a_2^0 \end{bmatrix} = \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad (1)$$

$$\mathbf{z}^1 = [z_1^1] \quad (2)$$

$$\mathbf{b}^1 = [b_1^1] \quad (3)$$

$$\mathbf{a}^1 = [a_1^1] \quad (4)$$

$$\mathbf{z}^2 = \begin{bmatrix} z_1^2 \\ z_2^2 \end{bmatrix} \quad (5)$$

$$\mathbf{b}^2 = \begin{bmatrix} b_1^2 \\ b_2^2 \end{bmatrix} \quad (6)$$

$$\mathbf{a}^2 = \begin{bmatrix} a_1^2 \\ a_2^2 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \mathbf{y} \quad (7)$$

$$\mathbf{W}^1 = \begin{bmatrix} W_{11}^1 \\ W_{21}^1 \end{bmatrix}, \mathbf{W}^2 = [W_{11}^2 \quad W_{12}^2] \quad (8)$$

Considerando esses dados as equações da propagação são dadas por

$$\mathbf{z}^1 = (\mathbf{W}^1)^T \cdot \mathbf{a}^0 + \mathbf{b}^1 \quad (9)$$

$$\mathbf{a}^1 = f_1(\mathbf{z}^1) = f_1(z_1^1) \quad (10)$$

$$\mathbf{z}^2 = (\mathbf{W}^2)^T \cdot \mathbf{a}^1 + \mathbf{b}^2 \quad (11)$$

$$\mathbf{a}^2 = f_2(\mathbf{Z}^2) = \begin{bmatrix} f_2(Z_1^2) \\ f_2(Z_2^2) \end{bmatrix} = \mathbf{y} \quad (12)$$

Com  $f_l$  sendo a função de ativação da camada  $l$ . Generalizando temos que:

$$\mathbf{Z}^l = (\mathbf{W}^l)^T \mathbf{a}^{l-1} + \mathbf{b}^l \quad (13)$$

$$\mathbf{a}^l = f_l(\mathbf{Z}^l) \quad (14)$$

Tendo  $\mathbf{Z}^l$ ,  $\mathbf{a}^l$  e  $\mathbf{b}^l$  as mesmas dimensões igual a  $n_l \times 1$  e  $\mathbf{W}^l$  de dimensões  $n_{l-1} \times n_l$  onde  $n_l$  representa o número de neurônios na camada  $l$ .

### Calculo de gradiente com propagação para trás (*Backpropagation*):

A propagação para trás está baseada na regra da cadeia do Cálculo diferencial e determina como calcular o gradiente de cada peso sináptico da rede neural, por exemplo determinando o custo na rede de exemplo mostrado na FIGURA 13 temos que para uma determinada iteração temos:

$$C_t = C_{y_1} + C_{y_2}$$

Assim tomando a derivada do erro total com respeito a um dos pesos obtemos o aporte do dito peso na variação do erro:

$$\frac{\partial C_t}{\partial W_{11}^1} = \frac{\partial C_{y_1}}{\partial W_{11}^1} + \frac{\partial C_{y_2}}{\partial W_{11}^1}$$

Logo por (7) temos que:

$$\frac{\partial C_{y_1}}{\partial W_{11}^1} = \frac{\partial C_{a_2^2}}{\partial W_{11}^1}$$

Aplicando a regra da cadeia e por (13) e (14) temos que:

$$\frac{\partial C_{a_2^2}}{\partial W_{11}^1} = \frac{\partial C_{a_2^2}}{\partial a_2^2} \frac{\partial a_2^2}{\partial Z_2^2} \frac{\partial Z_2^2}{\partial a_1^1} \frac{\partial a_1^1}{\partial Z_1^1} \frac{\partial Z_1^1}{\partial W_{11}^1}$$

Analogamente

$$\frac{\partial C_{a_1^2}}{\partial W_{11}^1} = \frac{\partial C_{a_1^2}}{\partial a_1^2} \frac{\partial a_1^2}{\partial Z_1^2} \frac{\partial Z_1^2}{\partial a_1^1} \frac{\partial a_1^1}{\partial Z_1^1} \frac{\partial Z_1^1}{\partial W_{11}^1}$$

Somando

$$\frac{\partial C_t}{\partial W_{11}^1} = \left( \frac{\partial C_{a_2^2}}{\partial a_2^2} \frac{\partial a_2^2}{\partial Z_2^2} \frac{\partial Z_2^2}{\partial a_1^1} \frac{\partial a_1^1}{\partial Z_1^1} \frac{\partial Z_1^1}{\partial W_{11}^1} + \frac{\partial C_{a_1^2}}{\partial a_1^2} \frac{\partial a_1^2}{\partial Z_1^2} \frac{\partial Z_1^2}{\partial a_1^1} \frac{\partial a_1^1}{\partial Z_1^1} \frac{\partial Z_1^1}{\partial W_{11}^1} \right) \quad (14)$$

Analogamente para  $b_1^1$  temos

$$\frac{\partial Ct}{\partial b_1^1} = \left( \frac{\partial C_{a_2^2}}{\partial a_2^2} \frac{\partial a_2^2}{\partial Z_2^2} \frac{\partial Z_2^2}{\partial a_1^1} \frac{\partial a_1^1}{\partial Z_1^1} \frac{\partial Z_1^1}{\partial b_1^1} + \frac{\partial C_{a_1^2}}{\partial a_1^2} \frac{\partial a_1^2}{\partial Z_1^2} \frac{\partial Z_1^2}{\partial a_1^1} \frac{\partial a_1^1}{\partial Z_1^1} \frac{\partial Z_1^1}{\partial b_1^1} \right) \quad (15)$$

Logo como  $\frac{\partial Z_j^l}{\partial a_k^{l-1}} = W_{kj}^l$ ,  $\frac{\partial Z_j^l}{\partial W_{kj}^l} = a_k^{l-1}$  é  $\frac{\partial Z_j^l}{\partial b_j^l} = 1$  por (13) temos que

$$\frac{\partial Ct}{\partial W_{11}^1} = \left( \frac{\partial C_{a_2^2}}{\partial a_2^2} \frac{\partial a_2^2}{\partial Z_2^2} W_{12}^2 \frac{\partial a_1^1}{\partial Z_1^1} a_1^0 + \frac{\partial C_{a_1^2}}{\partial a_1^2} \frac{\partial a_1^2}{\partial Z_1^2} W_{11}^2 \frac{\partial a_1^1}{\partial Z_1^1} a_1^0 \right) \quad (16)$$

$$\frac{\partial Ct}{\partial b_1^1} = \left( \frac{\partial C_{a_2^2}}{\partial a_2^2} \frac{\partial a_2^2}{\partial Z_2^2} W_{12}^2 \frac{\partial a_1^1}{\partial Z_1^1} + \frac{\partial C_{a_1^2}}{\partial a_1^2} \frac{\partial a_1^2}{\partial Z_1^2} W_{11}^2 \frac{\partial a_1^1}{\partial Z_1^1} \right) \quad (17)$$

Tomando isto em conta e considerando  $*$  como o operador do produto elemento a elemento (*element-wise operation*) e  $da^l = \frac{\partial Ct}{\partial a^l}$ ,  $dZ^l = \frac{\partial Ct}{\partial Z^l}$  generalizamos a fórmula da seguinte forma:

Se  $l = L$

$$da^L = da^L = \nabla_y Ct = \nabla_{a^L} Ct \text{ com as mesmas dimensões que } a^L$$

Se não temos que

$$da^l = W^{l+1} dZ^{l+1}$$

Com

$$dZ^l = da^l * df_l(Z^l)$$

$$dW^l = a^{l-1} (dZ^l)^T$$

$$db^l = dZ^l$$

Com o operador  $\nabla$  sendo o Jacobiano da função com respeito ao vetor ou matriz de saída e definindo a derivada da função de ativação na camada  $l$  como

$$df_l(Z^l) = \begin{bmatrix} df_l(Z_1^l) \\ df_l(Z_2^l) \\ \vdots \\ df_l(Z_m^l) \end{bmatrix} = \begin{bmatrix} \frac{\partial a_1^l}{\partial Z_1^l} \\ \frac{\partial a_2^l}{\partial Z_2^l} \\ \vdots \\ \frac{\partial a_m^l}{\partial Z_m^l} \end{bmatrix}. \text{ Também por calculo matricial se obtém que}$$

$$\frac{\partial Z^l}{\partial W^l} = \frac{\partial ((W^l)^T a^{l-1} + b^l)}{\partial W^l} = (a^{l-1})^T \text{ e } dW^l = \frac{\partial Ct}{\partial W^l} = dZ^l (a^{l-1})^T$$

Sendo por conveniência da nossa notação utilizada a forma  $(dZ^l (a^{l-1})^T)^T = a^{l-1} (dZ^l)^T$ . O processo é análogo para  $db^l$ .

No caso de utilizar lotes para treinar a rede se tem que  $(dW^l)_m = \frac{1}{m} \sum_m dW^l$  ou seja se obtém a média das gradientes dos pesos nos  $m$  dados do lote.

### Algoritmo de otimização

Para obter os novos valores dos pesos sinápticos após um treinamento e usado técnicas que visam a minimização do erro. Uma delas é a gradiente descendente (*GD*) mostrada a continuação:

$$W^l[i + 1] = W^l[i] - \alpha dW^l[i]$$

Onde  $\alpha$  é o hiperparâmetro que determina a progressão da aprendizagem.

Neste trabalho foi utilizado o algoritmo denominado ADAM (Estimação adaptativa do momento)(KINGMA; BA, 2014) que minimiza as iterações necessárias utilizando uma nova técnica que atualiza os momentos do GD, nela se tem que:

$$W^l[i + 1] = W^l[i] - \alpha \frac{\widehat{m}_{i+1}}{\sqrt{\widehat{v}_{i+1} + \varepsilon}}$$

$$\widehat{m}_{i+1} = \frac{m_{i+1}}{1 - \beta_1^{i+1}}, \widehat{v}_i = \frac{v_{i+1}}{1 - \beta_2^{i+1}}$$

$$m_{i+1} = \beta_1 \cdot m_i + (1 - \beta_1) dW^l[i]$$

$$v_{i+1} = \beta_2 \cdot m_i + (1 - \beta_2) dW^l[i]$$

$\beta_2, \beta_1$  são decaimentos exponenciais dos momentos de segunda e primeira ordem do gradiente e  $\varepsilon$  é uma pequena correção para evitar a divisão por zero. Ambos momentos são atualizados em cada iteração  $i$  do treinamento.

## ANEXO 2 – FUNÇÕES DE ATIVAÇÃO

As funções de ativação mais utilizadas na literatura em aplicações que envolvem a projeção das séries temporais são as seguintes:

- **Função Linear:**

$$\sum_{k=1}^n w_{kj}^l a_k^{l-1} + b_j = z_j^l$$

De acordo com (KUBRUSLY, 2011) a função linear é utilizada na saída da rede neural e é a indicada para se utilizar na saída das redes neurais quando a sequência de padrões de saída não são conjuntos limitados (TEIXEIRA, 2015).

- **Função Sigmoid:**

$$a_j^l = \frac{1}{1 + \exp(-z_j^l)} = \sigma(z_j^l)$$

A função logística sigmoide toma valores de [0,1], tendo a curva intermedia entre esses valores uma forma S.

- **Tangente Hiperbólica:**

$$a_j^l = \tanh(z_j^l)$$

A função tangente hiperbólica é uma função sigmoide com intervalo de variação [-1,1].

### ANEXO 3 – CÁLCULOS ESTATÍSTICOS E MATEMÁTICOS

**Cálculo do coeficiente de assimetria** segundo (KOKOSKA; ZWILLINGER, 2000):

$$g1 = \frac{m_3}{m_2^{\frac{3}{2}}}$$

Onde

$$m_r = \frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^r$$

É o r-ésimo momento com respeito à media

**Cálculo do coeficiente de curtose** segundo (KOKOSKA; ZWILLINGER, 2000):

$$g2 = \frac{m_4}{m_2^2}$$

**Calculo do Erro Absoluto Médio:**

$$MAE = \frac{1}{n} \sum_{j=1}^n |y_j - o_j|$$

**Calculo do Erro Porcentual Absoluto Médio:**

$$MAPE = \frac{1}{n} \sum_{j=1}^n \frac{|y_j - o_j|}{o_j}$$

**Calculo do Erro Quadrático Médio:**

$$MSE = \frac{1}{n} \sum_{j=1}^n (y_j - o_j)^2$$

Onde  $y_j$  representa a saída  $j$  do modelo,  $o_j$  o valor real na posição  $j$  e  $n$  a quantidade de saídas do modelo.

**Normalização utilizando a fórmula premnx:**

$$x' = \frac{x - \min(X)}{\max(X) - \min(X)}$$

Onde  $X$  é um vetor que contém todos os dados da série a ser normalizada a valores do intervalo  $[0,1]$ .

## APÊNDICE 1 – CÓDIGO IMPLEMENTADO

```

#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Mon Jul  9 16:55:30 2018

@author: walter
"""
#importacao das bibliotecas a serem usadas
from Modules.pre_processing_data import (feature_scaling,
                                         scaling_test)

import keras.backend as K
import pandas
import matplotlib.pyplot as plt
import copy
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM , Flatten
from keras.optimizers import Adam
from keras import regularizers
import openpyxl
import numpy as np
import tensorflow as tf
from numpy.random import import seed
from tensorflow import import set_random_seed
from random import import randint
from itertools import import zip_longest
from keras import import metrics

# Função que separa um array em grupos
def grouper(n, iterable, fillvalue=None):
    "grouper(3, 'ABCDEFG', 'x') --> ABC DEF Gxx"
    args = [iter(iterable)] * n
    return list(zip_longest(fillvalue=fillvalue, *args))

def train(y_train, output_dw, input_dw, n_epoch, n_batch, type_NN, nodes):
    # Gera um número randômico para a reprodutibilidade
    seed(random_number)
    set_random_seed(random_number)
    # A mistura dos dados(shuffle) e ativada pois se quer a recorrência
    somente na seqüência dos mesmos.
    shuf = True
    stateful_ = False
    x_train_input=[]
    y_train_output=[]
    # Define a quantidade de dados na sequencia temporal, neste caso só
    um dado por tempo.
    group_n= 1
    # Cria o modelo sequencial no keras
    model = Sequential()

    if (type_NN is "LSTM"):
        # Modela em caso que seja "LSTM"
        # Cria as sequencias de entrada com sequencias temporais de
        tamanho input_dw é as saídas com tamanho output_dw
        for i in range(0,len(y_train)-input_dw-output_dw + 1):
            x_train_input.append(grouper(group_n,y_train[i:i+input_dw]))

```

```

for i in range(input_dw, len(y_train)-output_dw + 1):
    y_train_output.append(y_train[i:i+output_dw])
y_train_output = np.array(y_train_output)
x_train_input = np.array(x_train_input)
    # Prepara o otimizador ADAM para o treinamento da rede
    optim = Adam(lr=0.0009, decay=0.5e-6)
    # Cria o tensor de entrada espacial-temporal do LSTM, sendo a forma
    # igual a quantidade de dados no lote, dimensão temporal, dimensão espacial
    x_train_input = x_train_input.reshape(x_train_input.shape[0],
x_train_input.shape[1],x_train_input.shape[2])
    # Criação da capa LSTM onde a quantidade de celas da mesma
    # depende exclusivamente da dimensão temporal do mesmo, com
    # return sequences=True a devolução de cada saída das celas e feita.
    # A regularização L1 é L2 e também definida no modelo.
    model.add(LSTM(nodes, batch_input_shape=(n_batch,
x_train_input.shape[1],x_train_input.shape[2]),
kernel_regularizer=regularizers.l1_l2(0.5e-6), activation='tanh',
stateful=stateful_, return_sequences=True))
    # Concatenação das saídas do modelo LSTM
    model.add(Flatten())
elif (type_NN is "MLP"):
    # Modela em caso que seja "MLP"
    # Cria as entradas e saídas para a rede MLP
    for i in range(0,len(y_train)-input_dw-output_dw + 1):
        x_train_input.append(y_train[i:i+input_dw])
    for i in range(input_dw, len(y_train)-output_dw + 1):
        y_train_output.append(y_train[i:i+output_dw])
    y_train_output = np.array(y_train_output)
    x_train_input = np.array(x_train_input)
    # Prepara o otimizador ADAM para o treinamento da rede
    optim = Adam(lr=0.00095, decay=1e-6)
    # Ativa o aleatoriedade da rede
    shuf = True
    model.add(Dense(nodes, activation='tanh', input_dim=input_dw,
kernel_regularizer=regularizers.l1_l2(0.5e-6)))
    # Saída linear para ambos modelos
    model.add(Dense(output_dw, activation='linear'))
    # Criação do compilador com a função objetivo
    model.compile(loss='mean_squared_error',
optimizer=optim, metrics=["mae"])

    # Criação dos array e vetores para armazenar as perdas e as
    # predições a serem feitas
    history = {}
    history["loss"]=[]
    history["val_loss"]=[]
    k = 0
    p_loss = 100
    m_loss = 100
    last_model = model.get_weights()
    x_prediction_input = grouper(group_n,list(y_train[-input_dw:]))
    scaled_y_test = np.divide(np.array(y_test)-scaled_y_train[2],
scaled_y_train[1]-scaled_y_train[2])
    scaled_val = np.array(scaled_y_test[-30:]).reshape(1,30)
    x_last_value = grouper(group_n,scaled_y_test[-input_dw:])
    if (type_NN is "LSTM"):
        x_prediction_input = np.array(x_prediction_input)
        print(x_prediction_input.shape)
        x_prediction_input = x_prediction_input.reshape(1,
x_prediction_input.shape[0],x_prediction_input.shape[1])
        x_last_value = np.array(x_last_value)

```

```

        x_last_value =
x_last_value.reshape(1,x_last_value.shape[0],x_last_value.shape[1])
    else:
        x_prediction_input =
np.array(x_prediction_input[1]).reshape(1,input_dw)
        x_last_value = np.array(x_last_value[1]).reshape(1,input_dw)
    #Inicio do treinamento
    #n epochs e a maxima quantidade de iteracao
    if (type_NN is "LSTM" or type_NN is "MLP"):
        print("\n:" + str(nodes) + ", i:" + str(input_dw) + ", o:" +
str(output_dw))
        for i in range(n_epoch):
            history_now = model.fit(x_train_input, y_train_output,
                                epochs=1, batch_size=n_batch,
                                validation_data=(x_prediction_input,
scaled_val), shuffle=shuf, verbose=False)
            history["loss"].append(history_now.history['loss'][-1])
            history["val_loss"].append(history_now.history['val_loss'][-1])
            # parada rápida com a verificação de que o erro vai diminuindo
            ao menos 1% em 10 iterações
            if history_now.history['loss'][-1] > 0.99*p_loss:
                k+=1
                print("k:"+str(k))
            else:
                p_loss = history_now.history['loss'][-1]
                k=0

            model.reset_states()
            if np.isnan(history_now.history['loss'][-1]):
                print("nan")
                model.set_weights(last_model)
                break
            else:
                aux = np.abs(history_now.history['mean_absolute_error'][-
1])*history_now.history['val_mean_absolute_error'][-1])
                if m_loss > aux:
                    # Se guarda os pesos do melhor modelo, eis
aquele no qual os erros são menores
                    last_model = model.get_weights()
                    m_loss = aux
                    mod_loss = history_now.history['mean_absolute_error'][-
1]

                    mod_val =
history_now.history['val_mean_absolute_error'][-1]
                    if k>10:
                        break
                    print(i)
                    # Se fica com o modelo com o menor erro no treinamento
                    model.set_weights(last_model)
                    history["loss"].append(mod_loss)
                    history["val_loss"].append(mod_val)
                return model, history, x_prediction_input, x_train_input,
y_train_output, x_last_value

if __name__ == '__main__':
# Numero de processadores utilizados para o modelado
    num_cores = 4
    GPU = False
    CPU = True
# Criação da semente para a aleatorização dos primeiros pesos sinápticos

```

```

random_number = int(np.random.random_sample()*5000)

if GPU:
    num_GPU = 1
    num_CPU = 1
if CPU:
    num_CPU = 1
    num_GPU = 0
# Configuração do tensorflow para funcionar com o CPU ou GPU
config = tf.ConfigProto(intra_op_parallelism_threads=num_cores,\
    inter_op_parallelism_threads=num_cores,\
allow_soft_placement=True,\
    device_count = {'CPU' : num_CPU, 'GPU' :
num_GPU})
session = tf.Session(config=config)
K.set_session(session)
name_xls = 'LSTM_2_F13Y_group_1.xlsx'
col = 0
# leitura dos dados
x = pandas.read_csv("/home/walter/Documents/Especializacion/Jairo-
Tesis/Dados_Seleccionados_JAIRO_recortado_win.csv", usecols=[col],
engine='python')
y_pred_data =
pandas.read_csv("/home/walter/Documents/Especializacion/Jairo-
Tesis/Resultados_LSTM_Puro/test_data.csv", usecols=[col], engine='python')
y_pred_data = y_pred_data.dropna(axis=0, how='any').values.tolist()
y_pred = []
for index, i in enumerate(y_pred_data):
    y_pred.append(float(i[0].replace(",",".")))
y_pred = np.array(y_pred)
x = x.dropna(axis=0, how='any').values.tolist()
set_test_size = 30
y = []
for index, i in enumerate(x):
    y.append(float(i[0].replace(",",".")))
y_train = y
# quantidade máxima de iterações (epochs) do treinamento se não funcionar a
parada rápida
n_epoch = 150
# quantidade de dados por lotes
n_batch = 1
ns = "_s1"
scaled_y_train = {}
# normalização dos dados com a formula premnmx [0,1]
scaled_y_train = feature_scaling(y_train, method='min-max')
scaled_y_train = (scaled_y_train[0][:set_test_size],
scaled_y_train[1], scaled_y_train[2])
# quantidade de hiperparâmetros a ser testado por uma busca por grade
input_dw_list = [5,10,15,20,25,30]
number_of_LSTM_units = [5,10,15,20,25,30]
output_dw_list = [30]
type_NN = "LSTM"
# Criação do arquivo excel para analise
workbook = openpyxl.load_workbook('Dados_Test.xlsx')
if ('Error_'+type_NN+ns) not in workbook.sheetnames:
    errorsheet = workbook.create_sheet('Error_'+type_NN+ns)
else:
    errorsheet = workbook['Error_'+type_NN+ns]

if ('Validation_Prediction'+type_NN+ns) not in workbook.sheetnames:
    prvsheet = workbook.create_sheet('Pred_RealValue_'+type_NN+ns)

```

```

else:
    prvsheet = workbook['Pred_RealValue_'+type_NN+ns]

    predictions_horizont = {}
    predictions_validation = {}
    error = {}
    models = {}
    val_error = {}
    test_error = {}
    mult_error = {}
# Treinamento dos distintos modelos
    for input_dw in input_dw_list:
        y_test = np.array(y[:-set_test_size-input_dw:])
        scaled_y_test = np.divide(np.array(y_test)-scaled_y_train[2],
scaled_y_train[1]-scaled_y_train[2])

        for output_dw in output_dw_list:
            for nodes in number_of_LSTM_units:
                model_fitted = 0
                (model_fitted, history, x_input, x_train_input,
y_train_output, x_last_input) = train(scaled_y_train[0], output_dw,
input_dw, n_epoch, n_batch, type_NN, nodes)
                # Previsão de validação e teste
                key = str(input_dw)+'_'+str(output_dw) + '_' + str(nodes)
                predictions_horizont[key] = []
                error[key] = [history['loss'], history['val_loss']]
                prediction =
model_fitted.predict(x_input)*(scaled_y_train[1]-scaled_y_train[2]) +
scaled_y_train[2]
                prediction_test =
model_fitted.predict(x_last_input)*(scaled_y_train[1]-scaled_y_train[2]) +
scaled_y_train[2]
                predictions_horizont[key] = prediction_test.tolist()[0]
                predictions_validation[key] = prediction.tolist()[0]
                val_error[key] = [np.mean(np.abs(y_test[:-set_test_size:] -
prediction)).tolist())
                test_error[key] = [np.mean(np.abs(y_pred -
prediction_test)).tolist())
                mult_error[key] = val_error[key]*np.array(error[key][0][-
1])*np.array(error[key][1][-1])
# Armazenamento dos dados no arquivo excel criado anteriormente
                col = 2
                errorsheet.cell(row=1, column=1).value = 'Type'
                errorsheet.cell(row=1, column=2).value = 'Train Error'
                errorsheet.cell(row=1, column=3).value = 'Validation Error on Training'
                errorsheet.cell(row=1, column=4).value = 'Validation Error'
                errorsheet.cell(row=1, column=5).value = 'Test Error'
                errorsheet.cell(row=1, column=6).value = 'Mult Error'
                errorsheet.cell(row=1, column=7).value = str(random_number)

    for key in error.keys():
        row_ = 1
        errorsheet.cell(row=col, column=row_).value = key
        for item in error[key]:
            row_ +=1
            errorsheet.cell(row=col, column=row_).value = item[-1]
        col += 1
    col = 2
    for key in val_error.keys():
        row_ = 3
        for item in val_error[key]:

```

```

        row_ +=1
        errorsheet.cell(row=col, column=row_).value = item
    col += 1
col = 2
for key in test_error.keys():
    row_ = 4
    for item in test_error[key]:
        row_ +=1
        errorsheet.cell(row=col, column=row_).value = item
    col += 1
col = 2
for key in mult_error.keys():
    row_ = 5
    for item in mult_error[key]:
        row_ +=1
        errorsheet.cell(row=col, column=row_).value = item
    col += 1

col = 1
for key in predictions_validation.keys():
    row_ = 1
    prvsheet.cell(row=row_, column=col).value = key + "_real"
    for item in y[-set_test_size:]:
        row_ +=1
        prvsheet.cell(row=row_, column=col).value = item
    row_ = 1
    prvsheet.cell(row=row_, column=col+1).value = key + "_val"
    for item in predictions_validation[key]:
        row_ +=1
        prvsheet.cell(row=row_, column=col+1).value = item
    row_ = 1
    prvsheet.cell(row=row_, column=col+2).value = key + "_test"
    for item in predictions_horizont[key]:
        row_ +=1
        prvsheet.cell(row=row_, column=col+2).value = item

    col += 3
# Guardando o arquivo na pasta definida por name_xls
workbook.save(name_xls)

```