INVESTIGANDO UMA SOLUÇÃO PARA PREVISÃO DE FALHAS EM TURBINAS HIDRÁULICAS POR MEIO DE DETECÇÃO DE ANOMALIA, *DEEP LEARNING* E A ARQUITETURA *AUTOENCODER*

Karen Assunção de Farias (IC) e Orlando Bisacchi Coelho (Orientador)

Apoio: PIBIC MackPesquisa

RESUMO

Esta pesquisa apresenta uma solução baseada em redes neurais da classe *deep learning*, utilizando a arquitetura *Autoencoder*, para solucionar o problema de manutenção preditiva de falhas em turbinas hidráulicas. Foi utilizada uma base de dados fornecida por uma operadora de energia do estado de São Paulo, contendo dados, coletados por meio de sensores, relativos ao funcionamento de um equipamento e os alarmes gerados pelo sistema supervisório a partir desses dados. Foram identificados quais desses eram mais influentes para o contexto do problema. Em seguida, foi desenvolvido um modelo computacional capaz de prever falhas futuras dessa turbina, o que possibilita a realização de manutenções preditivas. Abordamos a questão sob o paradigma de Detecção de Anomalias. Nesse sentido, optamos por uma rede *deep learning Autoencoder*. Para implementá-la, usamos o TensorFlow 2, uma biblioteca *deep learning* que pode ser chamada a partir de *scripts* em Python.

Palavras-chave: Manutenção Preditiva, Detecção de Anomalia, *Deep Learning*, *Autoencoder*, Turbinas Hidráulicas.

ABSTRACT

This research presents a solution based on Deep Learning, using the Autoencoder architecture, to solve the problem of predictive maintenance for hydraulic turbines. A dataset was provided by an energy operator in the state of São Paulo, containing data related to the operation of the equipment, collected through sensors, and the alarms generated from the data by the plant's supervisory system. First it was identified which data were more influential in the context of the problem. Then a computational model capable of predicting possible future failures of these turbines was developed, which makes it possible to carry out predictive maintenance. We approached the issue as an Anomaly Detection problem. In this sense, we chose a deep learning network of the Autoencoder class. To implement it, we used TensorFlow 2, a deep learning library that can be called from Python scripts.

Keywords: Predictive Maintenance, Anomaly Detection, *Deep Learning*, *Autoencoder,* Hydraulic Turbines.

1. INTRODUÇÃO

Os equipamentos utilizados em usinas hidráulicas são considerados um investimento de grande porte e eles são utilizados para a realização de atividades essenciais à sociedade: a geração de energia elétrica (VALLIM FILHO et al., 2019). Desta maneira, a realização da manutenção dessas turbinas é importante, de forma a otimizar o alto investimento e possibilitar a previsão de possíveis falhas, evitando interrupções desnecessárias no funcionamento do equipamento. A pesquisa reportada neste artigo visa contribuir para aprofundar o conhecimento e uso da manutenção preditiva das turbinas para alcançar seu maior desempenho, maximizar a entrega de energia elétrica a sociedade, mitigar o risco de interrupção de funcionamento dos equipamentos, reduzir os custos e preservar o alto investimento.

Dentre os paradigmas de manutenção, este projeto tem como base de desenvolvimento a manutenção preditiva (FU et al., 2004). Esse paradigma tem como finalidade auxiliar na detecção das condições de funcionamento de uma máquina, com base em dados extraídos e analisados, de forma a permitir a criação de modelos que emitam avisos que prevejam possíveis pontos de falha. A manutenção preditiva permite prever o acontecimento de possíveis falhas nas turbinas hidroelétricas, indicando o momento em que seja necessário realizar a manutenção, de forma a evitar a interrupção da operação dos equipamentos antes do tempo necessário.

Em conjunto com uma operadora de energia do estado de São Paulo, a Universidade Presbiteriana Mackenzie desenvolveu um projeto, cujo objetivo final foi a geração de um modelo de predição de falhas dos equipamentos geradores de energia elétrica. Para o desenvolvimento do projeto foram utilizadas uma turbina do complexo gerador da operadora (VALIM FILHO et al., 2019). A partir dos dados coletados dessa turbina e de alarmes gerados no sistema supervisório dos equipamentos, foi realizada a análise para descoberta dos fatores mais influentes e críticos (externos/ambientais e operacionais) em cenários de falha e/ou parada dos equipamentos.

O principal objetivo deste trabalho foi, a partir da análise e processamento desses dados gerados de sensores e alarmes, desenvolver um modelo preditivo empregando o paradigma de *Deep Learning*, um ramo da aprendizagem de máquina baseado na aprendizagem de representações (CHOLLET, 2018). Mais especificamente, a arquitetura usada é a *Autoencoder* (RUMELHART; HINTON; WILLIAMS,1986), um tipo de rede utilizado para o aprendizado de representação dos dados de forma não supervisionada. A tarefa de Aprendizagem de Máquina explorada é a Detecção de Anomalia (CHANDOLA; BANERJEE; JUMAR, 2009). Isto é, as eventuais falhas na operação são encaradas como anomalias.

De modo a contemplar esse objetivo, foi primeiramente estudado o funcionamento das turbinas, abrangendo o conhecimento de seus elementos constituintes e o ambiente em que elas operam. A seguir, estudamos como modelar o problema específico exatamente como uma tarefa de detecção de anomalia. Em seguida, estudamos a rede Autoencoder como a arquitetura *deep learning* mais apropriada para detecção de anomalia. Por fim, foi realizada a criação do modelo, sua execução com os dados extraídos dos equipamentos e a análise dos resultados obtidos.

A seção 2, a seguir, apresenta os principais conceitos teóricos usados na pesquisa. Na sessão 3 é apresentada a metodologia utilizada. Na sessão 4 são descritos e analisados os resultados obtidos a partir do modelo. Por fim, conclui-se a pesquisa com as discussões finais e possíveis trabalhos futuros.

2. REFERENCIAL TEÓRICO

Nesta seção, são tratados os temas necessários para a compreensão do trabalho, essenciais para a compreensão do modelo computacional desenvolvido. Os temas contemplados são: Manutenção Preditiva, Detecção de Anomalia, Aprendizagem de Máquina, *Deep Learning* e a rede *Autoencoder*.

2.1 Manutenção Preditiva

A manutenção é um fator importante para se maximizar o funcionamento de equipamentos e existem diversas formas de se realizar manutenção numa planta industrial, incluindo a manutenção preditiva.

Na manutenção preditiva (FU et al., 2004), a manutenção é realizada a partir de modelos matemático-computacionais baseada no estado do equipamento, permitindo a previsão de possíveis falhas futuras. São extraídos dados referentes ao funcionamento dos equipamentos ao longo do tempo, dos quais é possível identificar os estados de falha, e esses dados são utilizados no treinamento dos modelos computacionais. Esses modelos permitem prever o momento em que será necessária a interrupção do funcionamento do equipamento, evitando uma parada em produção desnecessária (POOJA; REKHA, 2020). Com isso, maximiza-se a disponibilidade dos equipamentos com um bom desempenho e os investimentos realizados, diminuindo os custos de manutenção e evitando interrupções não planejadas.

2.2 Detecção de Anomalia

Anomalias (CHANDOLA; BANERJEE; JUMAR, 2009) são padrões encontrados nos conjuntos de dados que não representam seu comportamento normal e podem ser inseridas nos dados por diferentes motivos, geralmente se referindo a problemas como fraudes, defeitos, dentre outros. As anomalias também podem ser de diferentes tipos, como anomalias coletivas, onde um conjunto de dados é anômalo, e anomalias contextuais, quando dados são considerados anômalos em determinado contexto, por exemplo.

A detecção de anomalia se refere ao problema de realizar a identificação dos dados que apresentam comportamento anômalo e que, em geral, representam comportamentos críticos dos sistemas. As técnicas de análise dos dados e detecção de anomalia podem envolver várias abordagens baseadas em Aprendizagem de Máquina, como agrupamento e classificação, entre outras (AGRAWAL; AGRAWAL, 2015).

O problema da manutenção preditiva pode ser entendido como uma tarefa de detecção de anomalia, visto que o comportamento normal do sistema é o esperado, e a falha é considerada a anomalia.

2.3. Aprendizagem de Máquina

A Aprendizagem de Máquina (GERON, 2019) é um subcampo da Inteligência Artificial onde são desenvolvidos algoritmos capazes de criar modelos a partir de dados que são fornecidos a eles, possibilitando aos computadores realizar funções para as quais não tenham sido diretamente programados. São apresentados dados de entrada (e, às vezes, também dados da respectiva saída pretendida) como exemplos para esses algoritmos e, a partir desses dados, é realizado o aprendizado do sistema, de forma a gerar a saída após o treinamento.

A tarefa de aprendizado de máquina pode ser classificada em três categorias: aprendizado supervisionado, onde são apresentadas as saídas esperadas pelo sistema para as entradas fornecidas no treinamento, aprendizado não supervisionado, no qual o algoritmo desconhece os dados de saída que se deseja que ele produza, e aprendizado por reforço, onde são enviados *feedbacks* binários ao sistema de acordo com a saída produzida (CHOLLET, 2018).

O uso de aprendizagem de máquina simplifica a realização de tarefas com muitas regras, simplificando a geração de códigos, auxilia na criação de sistemas onde é necessária constante adaptação e permite a realização de tarefas para as quais não dispomos de nenhum algoritmo eficiente.

2.4 *Deep Learning*

Como um subcampo da Aprendizagem de Máquina, temos *Deep Learning* (CHOLLET, 2018): uma abordagem que privilegia o aprendizado de representações para os dados (LE CUN; BENGIO; HINTON, 2015). Nessa abordagem, os modelos são compostos por camadas constituídas por unidades computacionais fracamente inspiradas pelo neurônio biológico. Essas unidades estão ligadas entre si por conexões às quais são associados pesos modificáveis, que podem ser tanto positivos como negativos. A partir das entradas, essas camadas abstraem as informações mais importantes e passam esse resultado para as próximas camadas. A ativação de cada unidade é calculada a partir das ativações das unidades que se conectam a ela, moduladas pelos pesos entre essas conexões. A ativação da unidade é uma função não linear da informação que ela recebe. Essa ativação é passada às unidades nas camadas seguintes da rede, e assim sucessivamente. A partir de um conjunto de dados de entrada, uma saída é então gerada, como sendo a ativação dos neurônios de saída.

A função de perda compreende à distância entre a saída gerada pelo modelo e o objetivo esperado, a fim de medir o erro produzido pela rede para uma dada entrada. Essa distância é utilizada como um *feedback* pelo algoritmo *backpropagation.* O algoritmo *backpropagation* (RUMELHART; HINTON; WILLIAMS,1986), principal algoritmo de *Deep Learning*, desempenha o trabalho de otimizador em uma rede *deep learning*. A partir da pontuação retornada pela função de perda, o algoritmo realiza ajustes nos pesos das conexões entre os neurônios de forma que minimize a diferença entre o resultado da rede e o objetivo esperado. Ao final do treinamento, os pesos representam o conhecimento adquirido pela rede sobre como realizar a tarefa para a qual ela foi treinada.

2.5 Rede Neural Recorrente LSTM

As redes neurais recorrentes (RNNs) (HOCHREITER; SCHMIDHUBER, 1997), diferente das redes tradicionais onde as informações não são persistidas, têm sua natureza em cadeia, funcionando com *loops*, de forma que as informações possam ser passadas para camadas anteriores da rede. Com isso, espera-se gerar a “memória” das informações passadas, alcançando uma melhor saída para o sistema. Entretanto, conforme a distância entre as camadas vai aumentando, a capacidade das RNNs para conectar a informação presente com a informação passada diminui.

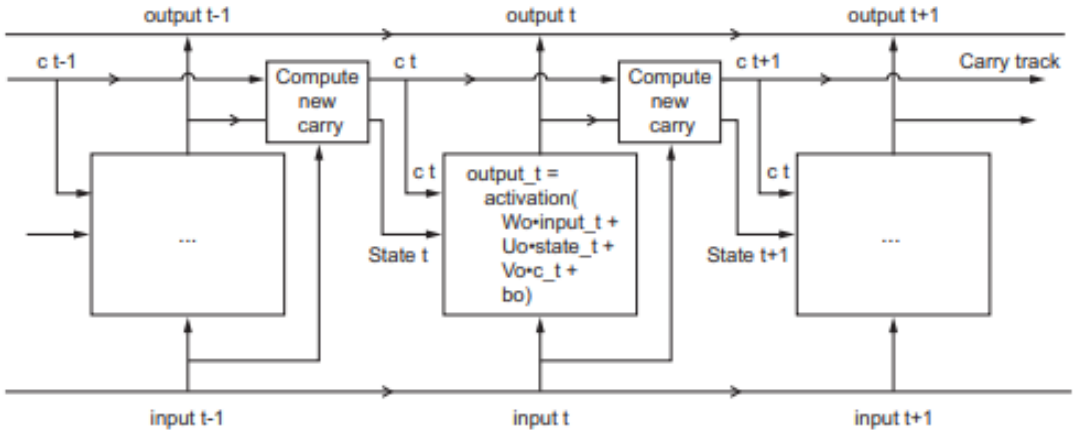
As redes Long Short Term-Memory, também chamadas LSTMs, são um tipo de RNN capaz de aprender, de forma melhor que as outras RNNs, a dependência entre as informações

a longo prazo. Por isso, são mais apropriadas quando o comportamento da informação ao longo do tempo é relevante para o modelo.

Essas redes também seguem a estrutura em cadeia das RNNs, mas diferem em sua camada (ou camadas) responsável pelo estabelecimento da “memória da rede”. Essa camada contém células LSTM bastante complexas, as quais contém quatro portas cuja função é modular a interação entre informação presente e passada. Em uma das portas, a rede decide qual informação deve ser esquecida, em outra decide-se qual deve ser adicionada, em outra camada decide-se se como alterar o estado corrente da célula e, na última, decide-se a saída a ser produzida pela célula LSTM. Esse processamento é ilustrado na Figura 1.

Figura 1. Estrutura Básica de uma rede *LSTM*.

Retirada de [Chollet, 2018].



A maior desvantagem da rede LSTM é o fato de o processamento das células LSTM não ser paralelizável. Para fazer face a essa limitação foi desenvolvida recentemente a arquitetura *Transformer* (VASWANI et al., 2017).

2.6 *Autoencoder*

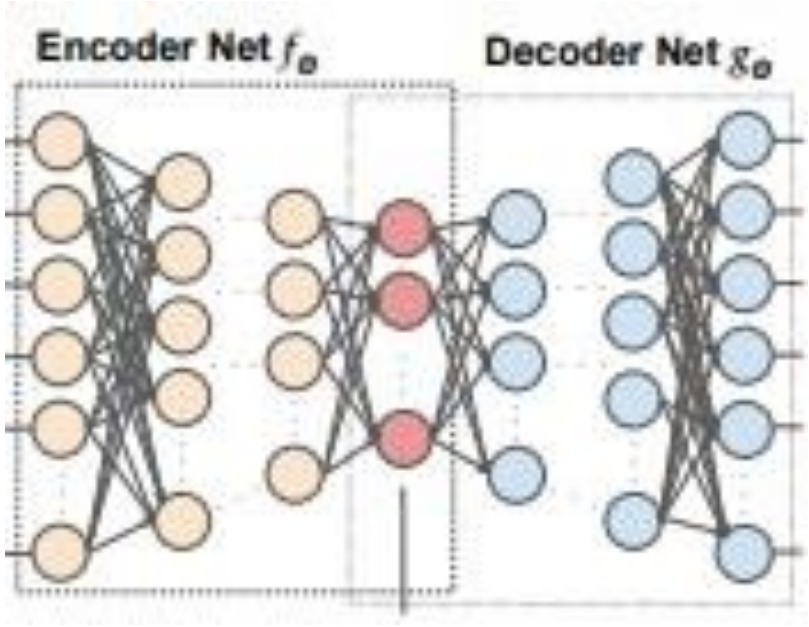
Autoencoder (RUMELHART; HINTON; WILLIAMS, 1986; CHREYER, M. et al., 2018) é uma rede *deep learning* do tipo *feedforward*, uma rede onde não existem ligações entre neurônios artificiais na mesma camada, ou de um neurônio artificial para um outro em camada anterior. Essa rede *Autoencoder* pretende aprender a produzir representações de maneira não supervisionada, sendo treinada na tarefa de auto-associação, reproduzindo na camada de saída os valores da camada de entrada.

Nessa rede as camadas escondidas apresentam redução de tamanho em número de unidades à medida que se afasta da camada de entrada, até um certo tamanho mínimo. A partir daí as camadas vão crescendo, na mesma proporção, até a camada de saída. Com isso, a dimensionalidade dos dados de entrada é reduzida: a entrada é codificada em um espaço latente, de menor dimensão e, nas camadas seguintes, é realizada a decodificação gerando a reconstrução para a saída da rede. Para cada entrada, a rede *Autoencoder* busca reproduzir o mesmo padrão na saída a partir dessa representação em espaço latente (GONDARA, 2016).

A arquitetura *Autoencoder* é baseada em duas partes principais: o codificador (*encoder*) e o decodificador (*decoder*). O codificador é responsável pela compactação da entrada em uma representação de espaço latente. Essa representação, por ser a ativação das unidades na camada escondida de dimensão mais reduzida, tende a capturar as características mais essenciais dos dados de entrada, de forma que seja capaz de reproduzir fielmente as saídas. O decodificador é responsável pela reconstrução dos dados a partir da representação do espaço latente, gerando a saída da rede o mais fiel possível, como é apresentado na Figura 2 a seguir.

Figura 2. Estrutura básica de uma rede *Autoencoder*.

Adaptada de [Schreyer, 2018].



Como citado na seção 2.4, a função de perda é utilizada para calcular a distância entre o resultado obtido e o resultado esperado. No caso da rede *Autoencoder* essa distância é

calculada com base nos dados de entrada, buscando-se gerar dados de saída o mais próximo possível dos dados de entrada.

3. METODOLOGIA

Nessa seção é descrito o processo de construção da base de dados utilizada no experimento e o desenvolvimento do modelo proposto neste trabalho.

3.1 Base de dados

Para a construção da base de dados utilizada no treinamento do experimento foi gerado um conjunto com os registros iniciais das séries temporais dos alarmes. Para isso, utilizamos os dados dos alarmes disponíveis de 17/05/2018 a 19/10/2019, dos quais foram considerados os alarmes no período de 08/01/2019, às 10:42:42.047, a 14/02/2019, às 16:30:57.510.

Os dados contemplam a marca temporal de cada alarme, momento em que o alarme é acionado, apresentada na coluna *E3TimeStamp*, bem como sua nomenclatura técnica, código que determina o alarme que foi acionado, na coluna *Message*, como apresentado na Figura 3 a seguir.

Figura 3. Alarmes iniciais da base de treinamento.

E3TimeStamp	Message
2019-01-08 10:42:42.047	Comando Diminui Referência do Regulador de Velocidade
2019-01-08 11:51:33.667	Usina Externa - Excitação UG6 - Comando Subir Referência Acionado
2019-01-08 11:52:09.890	Usina Externa - Excitação UG6 - Comando Subir Referência Desacionado
2019-01-08 12:19:54.960	Comando Aumenta Referência do Regulador de Velocidade
2019-01-08 13:01:08.890	LIMITE SUPERIOR DE REF. DO LIMITADOR DE ABERTURA ATUADO
2019-01-08 13:02:08.840	DRENAGEM PORÃO - NÍVEL LIGA BOMBA A
2019-01-08 13:10:53.877	DRENAGEM PORÃO - NÍVEL LIGA BOMBA B
2019-01-08 13:17:39.663	Comando Aumenta Referência do Regulador de Velocidade
2019-01-08 13:23:42.973	DRENAGEM PORÃO - NÍVEL LIGA BOMBA A
2019-01-08 13:24:43.907	DRENAGEM PORÃO - NÍVEL LIGA BOMBA B

Embora a base possa apresentar outros alarmes que sinalizem algum evento indesejado nas turbinas, os alarmes que representam algum tipo de falha grave, ou a parada do funcionamento dos equipamentos, são os alarmes com o código “86”, fato esse que nos foi reportado pelos especialistas que atuam com as turbinas. A base de treinamento

considerada para o experimento atual apresenta 4 alarmes com código “86”, que são apresentados na Figura 4 a seguir.

Figura 4. Alarmes com código “86” presentes na base de treinamento.

E3TimeStamp	Message
2019-01-09 15:16:31.373	Usina Externa - UG6 - Prot. Principal - Relé de Bloqueio 86E - ATUADO
2019-01-29 15:33:17.230	Usina Externa - UG6 - Prot. Principal - Relé de Bloqueio 86E - ATUADO
2019-01-29 15:40:43.220	Usina Externa - UG6 - Prot. Principal - Relé de Bloqueio 86E - NORMALIZADO
2019-02-14 16:19:25.510	86E / 86H / 86M - OPERADO

Após a seleção do período em que ocorreram os alarmes, a base foi limpa de forma que ficassem para o treinamento apenas os alarmes iniciais das séries temporais, sem as consideradas “cascatas” de alarmes que ocorrem em sequência em um mesmo momento, incluindo sequências iguais de alarmes que são disparadas imediatamente, um após o outro. De fato, essas cascatas são subsequências de alarmes que são necessariamente disparados quando certos alarmes mais importantes são disparados; dessa forma, elas não configuram alarmes relevantes.

Uma consideração importante a fazer é que, embora haja um grande volume de alarmes, essa turbina apresenta baixa ocorrência de problemas que resultem em parada dos equipamentos, apresentando, então, poucas ocorrências de alarmes críticos do tipo 86. O que dificulta o processo de derivação de um modelo preditivo para as falhas nessa turbina, já que há poucas falhas.

3.2 Criação do conjunto de treinamento

Para a criação do conjunto de treinamento para a entrada da rede, foram considerados 1.831 alarmes no período citado anteriormente. Nesse conjunto, há um total de 189 alarmes distintos, bem como suas eventuais repetições. Em seguida, com a utilização de um dicionário, estrutura de dados na linguagem de programação Python que permite o mapeamento de dados em “chave-valor” demostrado na Figura 5, utilizamos o *one-hot encoding*, método que permite a transformação de dados em variáveis de forma binária, onde dados categóricos são transformados de forma que possuam um bit com a representação “1” e os demais são “0”, gerando vetores da dimensão do número de dados, vetores esses todos distintos entre si. Dessa forma, os alarmes foram codificados em vetores de dimensão 189.

Figura 5. Parte da estrutura de dicionário utilizada para a codificação dos alarmes.

Após a codificação, os alarmes foram distribuídos sequencialmente em 39 entradas distintas. Como a média diária de alarmes é 48 e a dimensão da representação de cada alarme é189 (o número de alarmes distintos), a dimensão de cada uma das 39 entradas ficou sendo 9.072 (números 0 ou 1). De forma que todas as entradas da rede tivessem a mesma dimensão, inclusive a última, utilizamos *padding* à direita com valores 0.5 para completar a dimensão da última entrada cujo tamanho era menor que as demais. As entradas foram então transformadas em tensores, tipo de dados necessário para a entrada da rede.

Do conjunto de dados obtido, foram separadas as entradas referentes aos alarmes cujo código possuía “86”, candidatos a anômalos, já que apontam para algum tipo de falha grave ou a parada do funcionamento dos equipamentos e, na mesma proporção, foram retiradas entradas cujos alarmes constituintes eram diferentes dos alarmes “86”.

Em seguida, o conjunto de dados resultante foi separado entre os dados de treinamento e dados de teste, resultando em 85% do conjunto de dados para treinamento e 15% para teste.

3.3 Modelo de rede *Autoencoder* utilizada

Após a construção do conjunto de dados a ser utilizado, foi criado o modelo de rede *Autoencoder*, utilizando TensorFlow 2, uma biblioteca de código aberto para o desenvolvimento de soluções de *machine learning*, que pode ser chamada a partir de *scripts* na linguagem de programação Python.

A rede foi definida com pares de camadas ocultas de dimensão 1.326, 194 e 28, como demonstrado na Figura 6 a seguir.

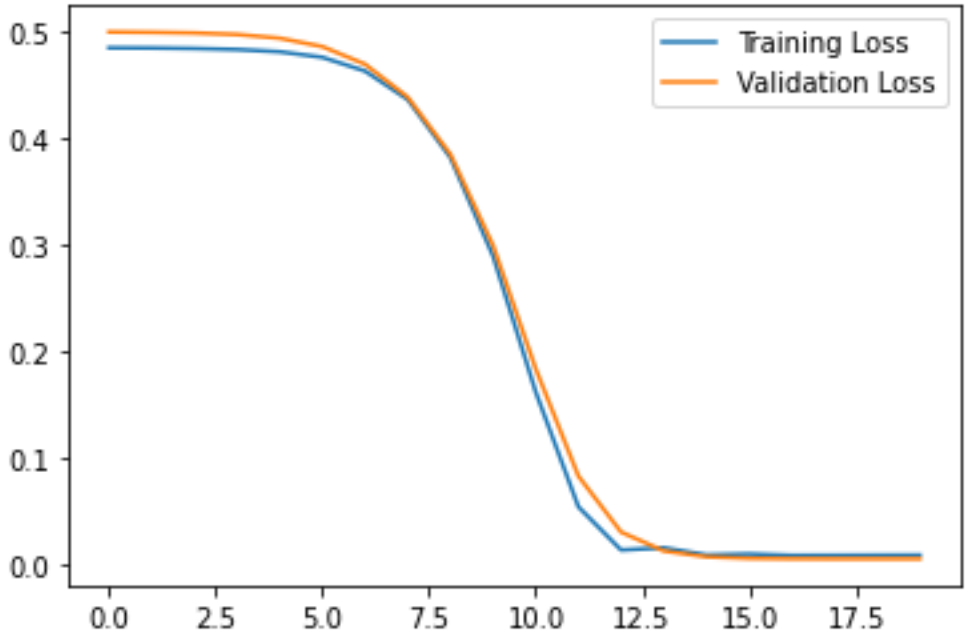
Figura 6. Rede *Autoencoder* utilizada no projeto.

```
[ ]  class AnomalyDetector(Model):
       def __init__(self):
         super(AnomalyDetector, self).__init__()
         self.encoder = tf.keras.Sequential([
           layers.Dense(1326, activation="relu"),
           layers.Dense(194, activation="relu"),
           layers.Dense(28, activation="relu")])

         self.decoder = tf.keras.Sequential([
           layers.Dense(194, activation="relu"),
           layers.Dense(1326, activation="relu"),
           layers.Dense(9072, activation="sigmoid")])

       def call(self, x):
         encoded = self.encoder(x)
         decoded = self.decoder(encoded)
         return decoded

     autoencoder = AnomalyDetector()
```

4. RESULTADOS E DISCUSSÃO

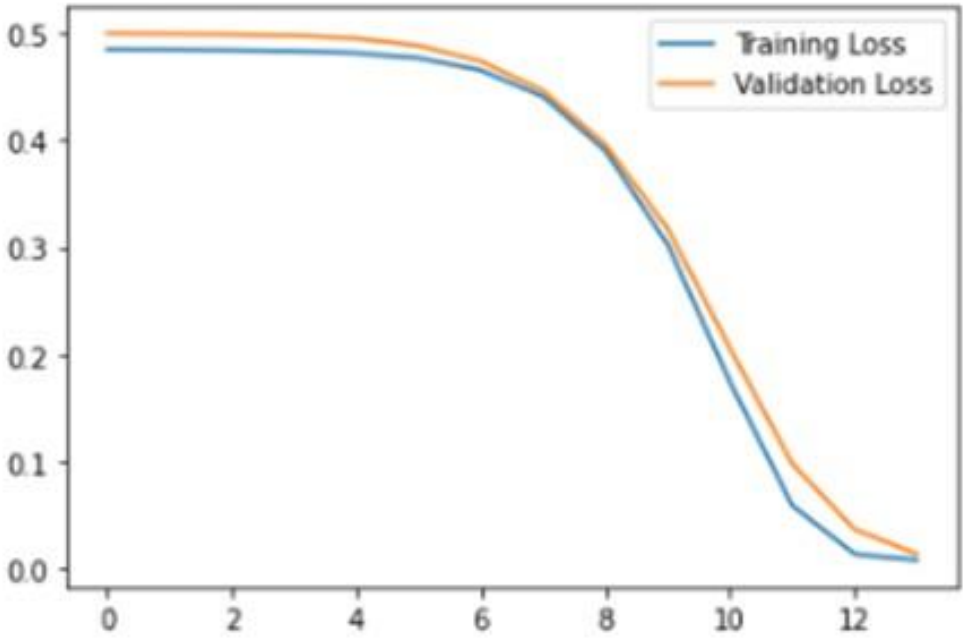
Foi realizado treinamento da rede por 20 épocas (onde uma época do treinamento corresponde à exibição de todo o conjunto de treinamento, sequencialmente, uma vez). Foi identificado experimentalmente que, após 14 épocas, a perda (loss) sobre o conjunto de treinamento se estabiliza enquanto a perda sobre o conjunto de validação ainda continua a cair, como mostrado na Figura 7.

Figura 7. Treinamento da rede por 20 épocas.



Então, a após treinada por 14 épocas, os pesos da rede foram congelados. A Figura 8 a seguir apresenta o momento em que as curvas se encontram.

Figura 8. Curvas se cruzando na época 14.



Após isso, testamos o modelo com o conjunto de dados “anômalos”, contendo os alarmes de código “86” e o conjunto de dados “normais”, não contendo os alarmes “86”. E verificamos que a perda sobre os dados anômalos é sensivelmente maior que a perda obtida sobre os dados normais, sendo a média da perda dos anômalos, de aproximadamente 0,053 e a média da perda do conjunto de dados normais de aproximadamente 0,017. O que comprova que a rede teve dificuldade de reconstruir as sequências de alarmes que continham os alarmes do tipo 86 -- os que indicavam um comportamento anômalo da turbina – significativamente maior do que a dificuldade que encontrou para reconstruir os dados que não correspondiam às anomalias. E isso não se pode atribuir tão somente à existência de alarmes do tipo 86 nessas entradas, já que essa é só uma das 48 entradas submetidas simultaneamente à rede, num dado momento. A dificuldade deve ser atribuída também ao contexto em que esses alarmes aparecem.

5. CONSIDERAÇÕES FINAIS

No trabalho aqui descrito desenvolvemos um modelo computacional, uma rede *Deep Learning,* capaz de prever possíveis falhas futuras nos equipamentos de turbinas hidráulicas, possibilitando assim sua manutenção preditiva.

Após a construção do modelo e sua execução utilizando dados dos alarmes fornecidos pela operadora de energia do estado de São Paulo, foi possível identificar que a rede teve maior dificuldade em reconstruir as sequências anômalas do que as sequências que não possuíam os alarmes referentes as falhas e/ou interrupções nos equipamentos. Dessa forma, a rede foi capaz de detectar sequências contendo alarmes críticas, diferenciando-as das que não tinhas tais alarmes.

Como proposta para a realização de trabalhos futuros, seria oportuno o desenvolvimento de um modelo a partir de dados provenientes de equipamentos que apresentem mais falhas e que, por isso, possibilitem um conjunto de dados com maior ocorrência de eventos anômalos.

6. REFERÊNCIAS

AGRAWAL, S; AGRAWAL, J. **Survey on Anomaly Detection using Data Mining Techniques**. ScienceDirect, 1 set. 2015. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1877050915023479>. Acesso em: 28 de maio de. 2021.

CHANDOLA, Varun; BANERJEE, Arindam; KUMAR, Vipin. Anomaly Detection: A Survey. **ACM Computing Surveys**, September, p. 1-72, 2009.

CHOLLET, F. **Deep Learning with Python**. New York: Manning Publications Co., 2018.

ELMAN, J. L. **Finding Structure in Time**. Cognitive Science 14(2), 1990.

VALLIM FILHO, A. R. A. et al. **Análise Preditiva Baseada em Inteligência Artificial para Sistemas Supervisórios de Usinas Hidráulicas**. 2019. Projeto de Pesquisa e Desenvolvimento (Faculdade de Computação e Informática) – Universidade Presbiteriana Mackenzie, São Paulo, 2019.

FU, Chuang; YE, Luqing; LIU, Yongqian; YU, Ren; IUNG, Benoit; CHENG, Yuanchu; ZENG, Yuming. Predictive Maintenance in Intelligent-Control-Maintenance-Management System for Hydroelectric Generating Unit. **IEEE Transactions on Energy Conversion**, v. 19(1), 2004.

GATELY, E. **Neural Networks for Financial Forecasting**. New York: John Wiley & Sons, 1995.

GERON, A. **Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow:** Concepts, Tools, and Techniques to Build Intelligent Systems.

GONDARA, L. **Medical image denoising using convolutional denoising autoencoders**. 2016. Disponível em:< https://arxiv.org/pdf/1608.04667.pdf>. Acesso em: 18 mai. 2021.

GROSSBERG; S, Mingolla, E. **A neural network architecture for preattentive vision**. IEEE Transactions on Biomedical Engineering, v. 36(1), p. 65-84, 1989.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. **Neural Computation** v. 9, p. 1735–1780, 1997.

LECUN, Yann; BENGIO, Yoshua; HINTON, Geoffrey E. Deep Learning. **Nature**, v. 521, p. 436-444, 2015.

LU, H.; SETIONO, R.; LIU, H. **Effective Data Mining Using Neural Networks**. IEEE Transactions on Knowledge and Data Engineering, v. 8(6), 1996.

POMERLEAU, D. A. Alvinn: **An Autonomous Land Vehicle in a Neural Network**. Technical Report, Carnegie-Mellon University, Pittsburgh, 1989.

POOJA, Kamat; REKHA, Sugandhi. **Anomaly Detection for Predictive Maintenance in Industry 4.0- A survey**. EDP Sciences, 28 mai. 2020. Disponível em:< https://www.e3s-conferences.org/articles/e3sconf/abs/2020/30/e3sconf_evf2020_02007/e3sconf_evf2020_02007.html>. Acesso em: 14 maio. 2021.

PYTHON. Python. https://www.python.org/. Acesso em 26 de julho de 2021.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. **Learning representations by back-propagating errors**. Nature, v. 323, p. 533–536,1986.

SCHREYER, M. et al. **Detection of Anomalies in Large-Scale Accounting Data using Deep Autoencoder Networks**. 2018. Disponível em:< https://arxiv.org/pdf/1709.05254.pdf>. Acesso em: 18 mai. 2021.

TENSORFLOW. TensorFlow. https://www.tensorflow.org/. Acesso em 26 de julho de 2021.

TESAURO, G. **Temporal Difference Learning and TD-Gammon**. Communications of the ACM, v. 38(3),1995.

VALENTIN, D. et al. **Connectionist models of face processing:** A survey. Pattern Recognition v. 7(9), p. 1209-1230, 1994.

VASWANI, A. et al. **Attention in all you need**. Cornell University. 06 dez. 2017. Disponível em: < https://arxiv.org/abs/1706.03762>. Acesso em: 11 jun. 2021.

WIDROW, Bernard; HOFF, Marcian E. **Adaptive Switching Circuits**. IRE WESCON Convention Record, Part 4. Proceedings, 1960, p. 96-104.

Contatos: karenaf2000@gmail.com e orlandobc@gmail.com