Sociedade Brasileira de Automática (SBA)
XV Simpósio Brasileiro de Automação Inteligente - SBAI 2021, 17 a 20 de outubro de 2021

ANÁLISE DE DESEMPENHO DE REDES NEURAIS LSTM COM TÉCNICAS DE *PRUNING* PARA DETECÇÃO DE FALHAS EM PROCESSOS INDUSTRIAS

Paulo Victor Correia[*], Lucileide Dantas[†], Luiz Affonso Guedes[*], Marcelo Fernandes[*]

[*] *Departamento de Engenharia de Computação e Automação*
Universidade Federal do Rio Grande do Norte
Natal, RN, Brasil

[†] *Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte*
Rua São Braz, 304, Bairro Paraíso
Santa Cruz, RN, Brasil

Emails: paulocorreiaufrn@gmail.com, lucileide.dantas@ifrn.edu.br, affonso@dca.ufrn.br, mfernandes@dca.ufrn.br

Abstract— In industry, real-time fault detection and diagnosis methods are required to secure processes, reduce damage to products, and avoid possible system failures. Recently, Long Short-Term Memory (LSTM) neural networks are used as an approach to fault detection in industrial process operation because of their strength sequential data processing, such as time series processing. However, LSTM neural networks demand more effort computational to inferring and training when compared to other kinds of neural network architectures. Then, considering IIoT (Industrial Internet of Things) embedded systems have limited memory capacity and small battery charges, strategies to speed up inference in LSTM neural networks and enhance their performance became necessary. In this way, this paper proposes a basis to compress LSTM neural networks with pruning techniques in software. Our pruning approach removes redundant parameters of the LSTM neural network by zeroing absolute synaptic weight values below a threshold. Then, we retrain the pruned model to readjust non-zero weights. We used the *Tennessee Eastman Process* benchmark to assess our approach. Finally, the paper presents the accuracy, precision, recall and F1-Score for both faulty data sets, varying the network's sparsity and comparing sparsities with performance parameters of the proposed network.

Keywords— *Long Short-Term Memory*, Anomaly Detection, Neural Network Compression, *Pruning*, *Tennessee Eastman Process*.

Resumo— Na indústria, métodos de detecção e diagnóstico de falhas em tempo real são necessários para proteger processos, reduzir danos aos produtos e evitar possíveis falhas do sistema. Recentemente, as redes neurais LSTM (*Long Short-Term Memory*) têm sido usadas como uma abordagem para detecção e diagnóstico de falhas na operação de processos industriais por causa de sua excelente capacidade de processar dados sequenciais, como processamento de séries temporais. No entanto, as redes neurais LSTM exigem mais esforço computacional para inferir e treinar seus modelos quando comparadas a outros tipos de arquiteturas de rede neural. Então, considerando que os sistemas embarcados de IIoT (*Industrial Internet of Things*) têm capacidades limitadas de processamento, memória, e fonte de energia, estratégias para acelerar a inferência em redes neurais LSTM e melhorar seu desempenho se tornaram bastante necessárias para viabilizarem sua aplicação em tais tipos de sistemas. Desta forma, este artigo se propõe a comprimir redes neurais LSTM com uma técnica de poda em *software*. A abordagem de poda utilizada remove os parâmetros redundantes da rede neural, zerando os valores absolutos de pesos sinápticos abaixo de um limite. Em seguida, os modelos podados são treinados novamente para reajustar pesos diferentes de zero. O conjunto de dados *benchmark Tennessee Eastman Process* será usado para avaliar nossa abordagem. Finalmente, serão apresentados resultados de taxa de acerto, precisão, sensibilidade e F1 para dois conjuntos de falhas diferentes, variando a esparsidade da rede e comparando esparsidade com parâmetros de performance da rede proposta.

Palavras-chave— *Long Short-Term Memory*, Detecção de Anomalias, Compressão de Redes Neurais, *Pruning*, *Tennessee Eastman Process*.

1 Introdução

No contexto da indústria 4.0, há a necessidade de monitoramento *online* de todas as etapas da operação de processos industriais. Algumas das principais atividades do monitoramento dessas operações são detecção, identificação, diagnóstico e prognóstico de falhas. Visando melhorar o desempenho dessas atividades, vem-se adotando cada vez mais abordagens orientadas a dados baseadas em redes neurais (Sun et al., 2020; Shenfield and Howarth, 2020). Sendo que mais recentemente alguns trabalhos indicam a adequação do uso de redes neurais do tipo LSTM (*Long Short-Term Memory*) devido à sua bem conhecida capacidade de processar dados sequenciais, como são as séries temporais das variáveis de processos industrias (de Oliveira et al., 2020; Jalayer et al., 2021). Porém, tais tipos de redes neurais demandam mais recursos computacionais de processamento e memória devido a sua estrutura em cascata dependente que causam gargalos para se realizar a operação de inferência (Wang et al., 2019; Gao et al., 2020). Além disto, em muitos sistemas industrias, como os baseados em IIoT (*Industrial Internet of Things*), tais recursos computacionais são bastante limitados, o que implica no emprego de técnicas baseadas em redes neurais de forma mais eficiente em termos computacionais, sem prejudicar seus desempenhos, para que sua adoção

seja viável.

Neste contexto, este artigo propõe a utilização da técnica de poda (*pruning*), que é uma técnicas de compressão, para tornar as redes neurais do tipo LSTM computacionalmente mais eficientes na fase de inferência sem comprometer os seus desempenhos. A aplicação de técnicas de compressão nas redes neurais LSTM se faz necessário devido à alta parametrização deste tipo de rede, que podem facilmente chegar a milhões de parâmetros (Kadetotad et al., 2020). A compressão destes modelos implica na redução de memória ocupada pelos mesmos (Wang et al., 2019) e no processamento da sua inferência.

Dito isso, este artigo propõe aplicar uma técnica de *pruning* apenas às redes neurais LSTM utilizadas como modelo de detecção de anomalias por meio do corte de pesos que tenham o valor absoluto menor que um limiar β. Também propõe-se uma estratégia de treinamento iterativo, o qual tem objetivo de aumentar a esparsidade enquanto mantém um desempenho satisfatório da rede. Ao final, obterá-se um conjunto de redes neurais LSTM comprimidas cujo desempenho será comparado ao de uma rede de referência sem compressão. Para validação e extração de resultados, será utilizado o simulador de processos *Tennessee Eastman Process* (TEP).

Alguns trabalhos na literatura propõem a utilização de técnicas de *pruning* para compressão de redes neurais LSTM. As técnicas mais comuns desta compressão aplicadas a esse tipo de rede são podas aleatórias de seus parâmetros (Kadetotad et al., 2020; Gil et al., 2021). Outra estratégia utilizada para compressão de redes é a utilização de Operadores de Produto de Matriz (MPO) para representação dos pesos sinápticos, o que reduz a quantidade de parâmetros armazenados (Gao et al., 2020). Há autores que também realizam *pruning* a partir análises de autovalores e médias geométricas dos pesos e eliminando as unidades mais redundantes da LSTM (Gkalelis and Mezaris, 2020).

O restante deste trabalho está dividido da seguinte forma: a Seção 2 apresentará a fundamentação teórica dos experimentos realizados; a Seção 3 revelará a metodologia utilizada no trabalho; a Seção 4 abordará as configurações dos experimentos realizados; a Seção 5 abordará os resultados e discussões para, finalmente, a Seção 6 apresentar as conclusões deste trabalho.

2 Redes Neurais LSTM

As redes neurais *Long Short-Term Memory* foram introduzidas em 1997, desenvolvidas para superar limitações presentes no treinamento das primeiras redes neurais recorrentes (Hochreiter and Schmidhuber, 1997). Estas limitações provocavam oscilações nos pesos durante o treinamento, sendo causadas pela explosão ou desaparecimento dos sinais de erros entre as unidades internas da rede. Isto comprometia a utilização destas redes em problemas sequenciais de longa dependência.

A Figura 1 apresenta a estrutura de uma célula LSTM. A arquitetura de uma célula LSTM consiste quatro perceptrons que compõem estruturas chamadas *gates*. O *forget gate* f_t determina informações de instantes passados que serão esquecidos pela célula por meio também de um neurônio ativado com uma função logística, cujo resultado será multiplicado ao estado da célula. O *input gate* i_t corresponde a uma ponderação das novas informações que serão introduzidas na rede a partir de um perceptron ativado com uma função logística. Ao mesmo tempo, estas informações também são ponderadas por um neurônio ativado porém com uma função tangente hiperbólica chamado *cell gate* g_t, cujo resultado será multiplicado com i_t e somado ao estado da célula. Finalmente, o *output gate* o_t calcula as informações que serão passadas para as próximas células e instantes de tempo seguintes (de Oliveira et al., 2020). Isto é feito aplicando uma função tangente hiperbólica ao estado da célula e multiplicando este resultado a o_t.

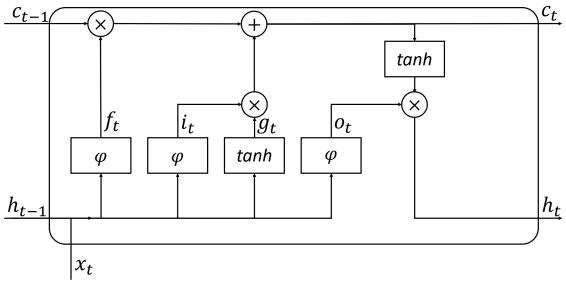

Figura 1: Diagrama da estrutura interna de uma célula LSTM.

Dito isso, cada célula LSTM possui um propagador interno de informação chamado *hidden state* h_t e um propagador de informações entre os instantes de tempo *cell state* c_t, os quais ambos são escalares. As saídas das células LSTM usadas para transmitir informações entre as camadas são os valores de h_t.

As equações (1) até (6) descrevem como os *gates* e os *states* são calculados para a inferência. As variáveis W_z com $z = \{i, f, g, o\}$ referem-se a matriz de pesos podáveis entre as entradas da rede LSTM e os *gates*. Já U_z refere-se às matrizes de peso entre os valores dos *hidden states* e os gates. Finalmente, b_z refere-se ao conjunto de *biases* presentes em cada gate das células da rede neural LSTM. x_t refere-se ao conjunto de entrada daquela célula para o instante de tempo t (Zhong et al., 2021). Finalmente

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \quad (1)$$

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \quad (2)$$

$$g_t = tanh(W_g x_t + U_g h_{t-1} + b_g) \quad (3)$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \quad (4)$$

$$c_t = f_t * c_{t-1} + i_t * g_t \quad (5)$$

$$h_t = o_t * tanh(c_t) \quad (6)$$

3 Compressão de Redes Neurais com Pruning

A estratégia de *pruning* para compressão de redes neurais consiste em remover conexões lineares entre as unidades do modelo em questão. Neste trabalho, o método adotado consiste em cortar pesos pequenos entre essas conexões a partir de um limiar pré-determinado β. O valor desta variável deve ser definido a partir da análise da distribuição dos pesos de cada uma das camadas. A equação (7) define como é feita esta operação de *pruning* em cada camada k. Nela, tem-se que $W_k(j)$ refere-se à cada elemento da matriz de pesos da camada em questão, assim como β_k define o limiar da camada.

$$P(W_k(j), \beta_k) = \begin{cases} W_k(j) & \text{se } |W_k(j)| \geq \beta_k \\ 0 & \text{se } |W_k(j)| < \beta_k \end{cases} \quad (7)$$

4 Metodologia

A estratégia de treinamento com compressão abordada neste artigo consiste em utilizar uma rede de referência para guiar os modelos comprimidos subsequentes para um mínimo local otimizado conhecido. Em seguida, é definido o valor do limiar β para cortar os pesos cujos valores absolutos das camadas LSTM que estejam abaixo. Por conseguinte, a rede é treinada outra vez com menos épocas para que os pesos remanescentes convirjam para um mínimo local próximo ao da rede guia. A seguir, aumenta-se o valor de β, cortam-se novamente os pesos que atendem à condição, e realiza-se novamente o treinamento da rede. O valor de β final é definido pelo usuário a partir de análise do desempenho para cada valor de β. Com isso, a compressão é realizada iterativamente no mesmo modelo após o retreinamento, produzindo cópias de *backup* para cada valor de β para análises futuras.

Esta forma de treinamento tem por finalidade utilizar uma rede que possui um bom desempenho como um guia para treinar modelos cada vez mais comprimidos com perca mínima de eficácia. A partir dos modelos obtidos, serão feitas análises em cada um deles para definir o que possui melhor desempenho com o máximo de compressão possível.

5 Configurações Experimentais

Esta seção elucidará as configurações experimentais do conjunto de dados utilizados neste artigo, bem como as configurações de treinamento das redes comprimidas. Além disso, descreverá as métricas de classificação utilizadas.

5.1 Tennessee Eastman Process

O conjunto de dados industriais utilizado é do *Tennessee Eastman Process*. Este é um processo simulado desenvolvido por Downs and Vogel (1993) e consiste na linha de produção que resulta em dois produtos e dois subprodutos. Esta operação utiliza os componentes A, B, C, D, E, F, G e H para realizar as reações mostradas nas Equações (8), (9), (10) e (11) .

$$A_{(g)} + C_{(g)} + D_{(g)} \rightarrow G_{(liq.)}, \quad \text{Produto 1} \quad (8)$$

$$A_{(g)} + C_{(g)} + E_{(g)} \rightarrow H_{(liq.)}, \quad \text{Produto 2} \quad (9)$$

$$3D_{(g)} \rightarrow 2F_{(liq.)}, \quad \text{Subproduto} \quad (10)$$

$$A_{(g)} + E_{(g)} \rightarrow F_{(liq.)}, \quad \text{Subproduto} \quad (11)$$

Nessa planta industrial há 41 variáveis para o acompanhamento do processo. Destas, 22 correspondem a valores contínuos diretamente obtidos de sensores acoplados aos componentes da planta. As outras 19 referem-se a composição dos componentes do processo em tubulações específicas (Downs and Vogel, 1993). Para este artigo, utilizaram-se as 22 variáveis contínuas de sensores em tempo real. Ademais, foi feito a normalização desse conjunto de dados para terem a média 0 e o desvio padrão 1.

Finalmente, foram realizadas duas simulações do TEP neste trabalho. O conjunto da primeira simulação serve para o treinamento tanto do modelo de referência quanto dos modelos comprimidos. Ela consiste em um conjunto de 17800 amostras com o distúrbio IDV8 aplicado à planta entre as amostras 12000 e 13000. Já a segunda simulação consiste de uma operação da planta com 15800 amostras e o distúrbio IDV1 aplicado entre as amostras 10000 e 11000. A Tabela 1 descreve os distúrbios IDV1 e IDV8 utilizados.

Tabela 1: Descrição dos distúrbios utilizados neste trabalho.

Distúrbio	Descrição
IDV1	Variação em degrau da razão do fluxo de alimentação dos componentes A/C com B constante.
IDV8	Variação aleatória no fluxo dos componentes A, B e C.

5.2 Treinamento das Redes Neurais LSTM Comprimidas

Este trabalho foi feito utilizando a linguagem de programação *Python* em conjunto com o *framework* de aprendizagem de máquina *Pytorch*. Devido a implementação ter sido feita apenas em software, não será feita a avaliação dos tempos de inferência das redes comprimidas. A rede neural LSTM usada como referência para a geração de modelos comprimidos possui uma arquitetura com duas camadas LSTM com 256 células em cada uma. Esta configuração de modelo resulta em 813056 parâmetros ajustáveis. A camada de saída do modelo de referência consiste em um perceptron ativado com a função logística, cujo objetivo é classificar o estado do sistema industrial no presente entre normal e anormal a partir das variáveis de processo.

A proposta de treinamento iterativo com compressão das redes LSTM por *pruning* utiliza um espaço linear entre 0 e 0,1 com 20 amostras igualmente distribuídas. Estes valores foram escolhidos após a análise das distribuições de todas as camadas LSTM do modelo de referência constatar uma distribuição próxima da uniforme. A distribuição destes valores possui um intervalo aproximado de -0,1 até 0,1. Logo, os valores de β escolhidos podam a maior parte dos parâmetros do modelo.

O algoritmo de ajuste de pesos utilizados neste trabalho tanto para o modelo de referência quanto para os subsequentes foi o Gradiente Descendente Estocástico. Este algoritmo treina o modelo de referência por 750 épocas, com taxa de aprendizagem de 0,005, e um *momentum* de 0,95. Já para o treinamento dos modelos comprimidos com *pruning* foi utilizado 100 épocas para cada um, taxa de aprendizagem de 0,01, e um *momentum* de 0,5, visando diminuir os efeitos de *overfitting*. O tamanho do mini-lote de treinamento utilizado é de 512 amostras.

5.3 Métricas de Avaliação

Este trabalho utiliza as métricas de classificação de taxa de acerto, precisão, sensibilidade e F1, as quais são definidas pelas Equações 12, 13, 14 e 15. Nessas equações, VP significa classificações Verdadeiro-Positivas, e VN significa Verdadeiro-Negativo, as quais representam a taxa de predições de classe que o modelo previu corretamente. Já FP significa Falso-Positivo e FN representa Falso-Negativo, os quais representam predições incorretas feitas pelo modelo de classificação. Neste caso de classificação binária, positivo refere-se à classe 1 e negativo à classe 0.

$$\text{Taxa de Acerto} = \frac{VP + VN}{VP + VN + FN + FP} \quad (12)$$

$$\text{Precisão} = \frac{VP}{VP + FP} \quad (13)$$

$$\textit{Sensibilidade} = \frac{VP}{VP + FN} \quad (14)$$

$$\textit{F1} = \frac{2 \cdot \textit{Sensibilidade} \cdot \text{Precisão}}{\textit{Sensibilidade} + \text{Precisão}} \quad (15)$$

6 Resultados e Discussões

Para avaliar o desempenho da abordagem proposta, utilizaremos o conjunto de dados industriais simulados *Tennessee Eastman Process* com um tipo de falha aplicada para o treinamento e outro tipo de falha nunca vista pelo modelo para a avaliação. Serão utilizadas as métricas de classificação de acurácia, precisão, sensibilidade e F1 para avaliar o desempenho em detecção de anomalias pelos modelos após cada retreinamento.

Finalmente, para avaliar a eficiência da compressão aplicada ao modelo, utilizaremos a métrica de esparsidade. Ela consiste na razão entre o número de pesos das camadas LSTM iguais a zero e o número total de pesos. Com isto, modelos de alta esparsidade terão menos parâmetros, reduzindo o espaço ocupado e potencialmente acelerando a inferência destes tipos de redes.

Esta seção dividirá a apresentação dos resultados para o conjunto de dados de treinamento com a falha IDV8, e de teste com a falha IDV1. Será feita a comparação do desempenho com as métricas de classificação dos modelos para cada valor de β e também a esparsidade obtida após cada realização de *pruning*.

6.1 Falha IDV8

A Figura 2 mostra o desempenho dos modelos para cada valor de β usado iterativamente para o treinamento. Nota-se que para todos os valores de β adotados, o método de treinamento manteve um alto desempenho constante. Para os parâmetros buscados, o modelo ótimo obteve o valor de 100% para todas as métricas de classificação utilizadas para o conjunto de treinamento.

Da mesma forma, a Figura 3 mostra os desempenhos dos modelos obtidos em relação a taxa de esparsidade de cada um. Observa-se que os modelo obtiveram taxas de esparsidade acima de 80% enquanto mantinham um desempenho ótimo na detecção de anomalias em todas as métricas para o conjunto de treinamento.

6.2 Falha IDV1

Agora, a Figura 4 apresenta o desempenho dos modelos para cada valor de β com o conjunto de dados de teste IDV1. Nela, verifica-se uma queda de desempenho do modelo em todas as métricas após $\beta = 0,045$. Aqui a taxa de acerto cai

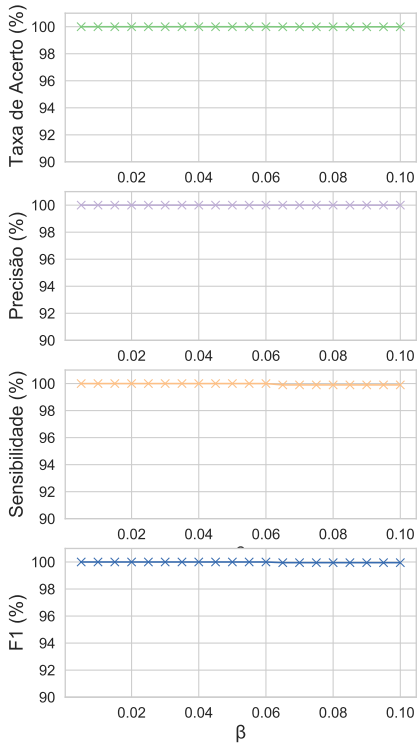

Figura 2: Resultados de desempenho das redes neurais LSTM comprimidas para cada valor de β para o conjunto de treinamento IDV8.

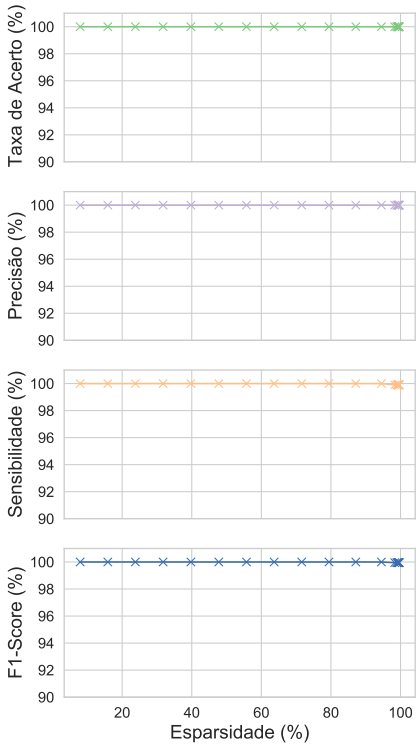

Figura 3: Gráfico comparativo entre esparsidade obtida pela metodologia proposta com o desempenho da rede em questão para o conjunto de treinamento IDV8.

de 97,5% para 95,4%, a precisão de 72,7% para 63,6%, a sensibilidade de 96,8% para 63,6%, e o F1 de 83,1% para 63,6%.

A Figura 5 apresenta um comparativo entre a esparsidade obtida para cada modelo e o desempenho de detecção de anomalias desconhecidas ao modelo. Neste caso da esparsidade, há uma queda no desempenho do modelo em relação ao conjunto de treinamento após atingir 71,6% de esparsidade. Esse valor foi obtida no modelo com $\beta = 0.045$, portanto a queda do desempenho observada na Figura 4 se manteve para a esparsidade.

Durante a realização deste trabalho, notou-se que a queda no desempenho ocorria em valores próximos ao desvio-padrão dos parâmetros das camadas LSTM pré-treinadas. Isto implica que a maior parte dos parâmetros relevantes para a inferência da rede não estão concentradas no centro da distribuição dos pesos das camadas LSTM. Portanto, a análise prévia dessas distribuições é necessária para aplicar a estratégia de treinamento a fim de obter modelos eficientes e eficazes.

Uma análise final dos resultados obtidos tanto de treinamento quanto de teste indica que a estratégia de treinamento resultou em modelos com *overfitting*. Constata-se esse fenômeno a partir da diferença significativa entre os desempenhos para o conjunto da falha IDV8 e IDV1. Apesar disso, a metodologia proposta obteve resultados satisfatórios também para o conjunto de dados de teste, obtendo desempenhos satisfatórios mesmo com uma redução de parâmetros de até 71,6% de esparsidade.

7 Conclusão

Este artigo propôs a utilização da estratégia de *pruning* para compressão de redes neurais LSTM durante seu treinamento. A estratégia de compressão consistia em podar os parâmetros da rede que tivessem valores absolutos abaixo de β, enquanto treinava um modelo a partir de uma rede neural LSTM de referência com cada vez menos parâmetros ajustáveis. Observa-se que o modelo manteve um desempenho satisfatório e constante para o conjunto de dados em que foi treinado. E mesmo com *overfitting*, obtivemos modelos de redes neurais LSTM comprimidos com esparsidades de até 71,7%, os quais tiveram métricas de acurácia, precisão, sensibilidade e F1 acima de 70% para dados desconhecidos.

Para trabalhos futuros, serão adotados estratégias estruturadas de *pruning*, com remoção aleatória de neurônios e parâmetros computados. Além disso, pretende-se utilizar estratégias de quantização para reduzir a largura de bits para representar os parâmetros utilizados numa rede neurais LSTM. Ademais novas estratégias de treinamento com balanceamento de dados, ampliação

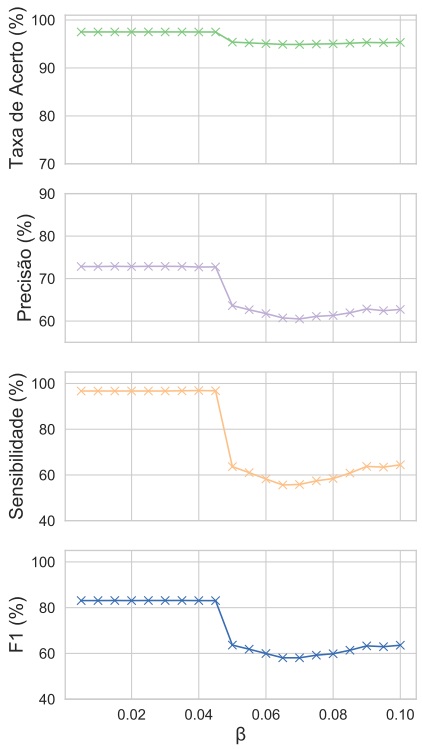

Figura 4: Resultados de desempenho das redes neurais LSTM comprimidas para cada valor de β para o conjunto de teste IDV1.

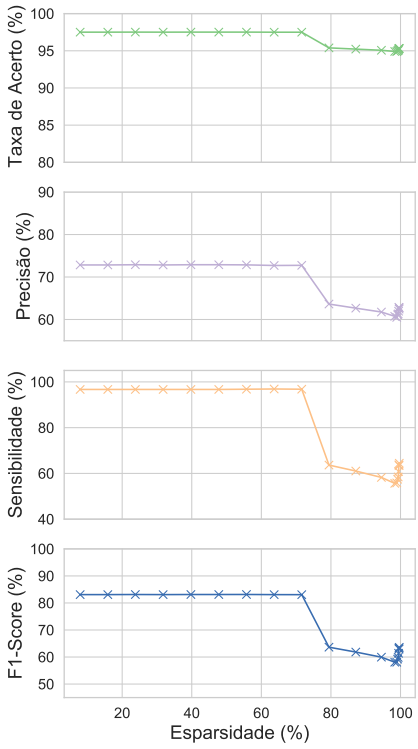

Figura 5: Comparativo entre esparsidade obtida pela metodologia proposta com o desempenho da rede em questão para o conjunto de teste IDV1.

da rede com camadas lineares, e diversificação das falhas aplicadas nos conjuntos de treinamento serão adotadas em trabalhos futuros. Finalmente, será feita a implementação dos modelos comprimidos em sistemas embarcados industriais, com o objetivo de melhorar o desempenho físico desses modelos em economia de energia, principalmente.

Referências

de Oliveira, E. V., Santos, M. R., Nunes, Y. T. P. and Guedes, L. A. (2020). Identificação de falhas de uma planta de nível piloto baseada em rede neural LSTM, *Anais do Congresso Brasileiro de Automática 2020*, sbabra.

Downs, J. and Vogel, E. (1993). A plant-wide industrial process control problem, *Computers & Chemical Engineering* **17**(3): 245–255.

Gao, Z.-F., Sun, X., Gao, L., Li, J. and Lu, Z.-Y. (2020). Compressing LSTM Networks by Matrix Product Operators.

Gil, Y., Park, J., Baek, J. and Han, S. (2021). Quantization-aware pruning criterion for industrial applications, *IEEE Transactions on Industrial Electronics* pp. 1–1.

Gkalelis, N. and Mezaris, V. (2020). Structured pruning of LSTMs via eigenanalysis and geometric median for mobile multimedia and deep learning applications, *2020 IEEE International Symposium on Multimedia (ISM)*, IEEE.

Hochreiter, S. and Schmidhuber, J. (1997). Long Short-Term Memory, *Neural Computation* **9**(8): 1735–1780.

Jalayer, M., Orsenigo, C. and Vercellis, C. (2021). Fault detection and diagnosis for rotating machinery: A model based on convolutional LSTM, fast fourier and continuous wavelet transforms, *Computers in Industry* **125**: 103378.

Kadetotad, D., Meng, J., Berisha, V., Chakrabarti, C. and sun Seo, J. (2020). Compressing LSTM networks with hierarchical coarse-grain sparsity, *Interspeech 2020*, ISCA.

Shenfield, A. and Howarth, M. (2020). A novel deep learning model for the detection and identification of rolling element-bearing faults, *Sensors* **20**(18): 5112.

Sun, W., Paiva, A. R., Xu, P., Sundaram, A. and Braatz, R. D. (2020). Fault detection and identification using bayesian recurrent neural networks, *Computers & Chemical Engineering* **141**: 106991.

Wang, S., Lin, P., Hu, R., Wang, H., He, J., Huang, Q. and Chang, S. (2019). Acceleration

of LSTM with structured pruning method on FPGA, *IEEE Access* **7**: 62930–62937.

Zhong, N., Li, Y. P., Li, X. Z., Guo, C. X. and Wu, T. (2021). Accurate prediction of salmon storage time using improved raman spectroscopy, *Journal of Food Engineering* **293**.