Identificação de Falhas em Sistemas UPS (Uninterruptible Power Supply) por meio de Deep Learning em Sistema Embarcado

Patrick R. P. Andrade[*] [▲]**
Gustavo Pessin[**]**

[*] *Programa de Pós-graduação em Instrumentação, Controle e Automação de Processos de Mineração, Universidade Federal de Ouro Preto (UFOP) e Instituto Tecnológico Vale (ITV), Ouro Preto, MG, Brasil (e-mail: patrick.andrade@aluno.itv.org)*
[**] *Instituto Tecnológico Vale, Ouro Preto, MG, Brasil (e-mail: gustavo.pessin@itv.org)*
[▲] *CEMIG Distribuição S.A., Belo Horizonte, MG, Brasil*

Abstract: The UPS system is extremely important for the Electric Power System (EPS), as it is responsible for ensuring monitoring and command in the case of lack of primary voltage. The three -phase rectifier is the most sensitive part of the UPS system and is the most susceptible to failures. Current rectifiers have an alarm system to indicate failures, but these alarms are most often coming late when the equipment has stopped working. This work proposes the development of an embedded solution using the Arduino Nano 33 BLE Sense and Deep Learning for Fault Identification in UPS systems through sound processing emitted by this equipment on a convolutionary neural network (CNN) using Google's Tensor Flow Lite Micro library. Results were obtained with 97.8% accuracy for identification of defective rectifiers.

Resumo: O sistema UPS é de extrema importância para o Sistema Elétrico de Potência (SEP), uma vez que é o responsável por garantir monitoramento e comando no caso da falta de tensão primária. O retificador trifásico é a parte mais sensível do sistema UPS e é o mais susceptível a falhas. Os retificadores atuais possuem um sistema de alarmes para indicar falhas, todavia esses alarmes, na maioria das vezes, vêm de forma tardia, quando o equipamento parou de funcionar. Esse trabalho propõe o desenvolvimento de uma solução embarcada utilizando a placa Arduino Nano 33 BLE Sense e Deep Learning para identificação de falhas em sistemas UPS através do processamento do som emitido por esses equipamentos em uma *convolutionary neural network* (CNN) utilizando a biblioteca Tensor Flow Lite Micro do Google. Foram obtidos resultados com acurácia de 97,8% para identificação de retificadores com defeito.

Keywords Failure Diagnoses in Industry; Electric Power System (EPS); Machine Learning; Deep Learning; Convolutionary Neural Networks (CNN); Tensor Flow Lite Micro; Sound Analysis; Spectogram.

Palavras-chaves: Diagnóstico de Falhas na Indústria; Sistema Elétrico de Potência (SEP); *Machine Learning*; *Deep Learning*; Redes Neurais Convolucionais (RNC); Tensor Flow Lite Micro; Análise de Som; Espectrograma.

1. INTRODUÇÃO

Em subestações de energia elétrica e usinas hidrelétricas, sistemas de fornecimento de energia ininterrupta (do inglês: *Uninterruptible Power Supply* – UPS) são partes essenciais para o funcionamento e segurança do Sistema Elétrico de Potência (SEP). Nessas instalações, todos os dispositivos de proteção, monitoração e controle do SEP funcionam em Corrente Contínua (CC), a qual é fornecida pelo sistema UPS, uma vez que a fonte primária de energia em Corrente Alternada (CA) não é uma opção durante faltas no SEP. Portanto, o sistema UPS é de extrema importância haja vista que sem ele seria impossível a abertura de disjuntores de alta tensão para interrupção de curtos-circuitos ou condições de risco à vida humana, como a queda de cabos ao solo. É graças ao sistema UPS que o Centro de Operação e Controle (COC) permanece com o monitoramento das instalações e pode operar os equipamentos de campo remotamente, e de forma ágil, para o restabelecimento de energia elétrica em caso de faltas.

O sistema UPS é composto basicamente por um retificador trifásico e um banco de baterias estacionárias que operam com tensão nominal de 125 VCC (tensão de corrente contínua). O retificador trifásico é a parte mais sensível do sistema UPS e é o mais susceptível a falhas, já que opera de forma ininterrupta fornecendo alimentação em corrente contínua para os equipamentos da instalação, bem como mantém o banco de baterias carregado. Atualmente o diagnóstico desse equipamento é fornecido através de alarmes do próprio equipamento, os quais chegam, muitas das vezes, quando o

equipamento já não está mais retificando, ou, no pior caso, não sobe alarme para o COC, o qual só toma conhecimento do problema quando perde totalmente o monitoramento e a possibilidade de comando.

A perda do retificador também afeta a proteção do SEP, uma vez que os relés de proteção operam em corrente contínua. Esse é sem dúvida o maior problema, uma vez que sem alimentação os relés não são capazes de identificar o curto-circuito, bem como ficam sem tensão para o sinal de Trip, responsável por abrir os disjuntores em caso de faltas no SEP. As consequências da não eliminação de curto-circuito podem ser catastróficas e acarretarem incêndios, morte de animais e seres humanos.

O diagnóstico de falhas em retificadores é mais eficiente quando realizado por um técnico experiente e treinado ao visitar a subestação de energia elétrica. O técnico treinado é capaz de reconhecer um retificador com falha ou na iminência de falhar por meio do som característico emitido pelo equipamento. Retificadores com problemas costumam emitir ruídos de alta frequência, os quais não estão presentes em retificadores em bom estado de funcionamento.

A experiência em campo mostra que retificadores emitem sons que vão se alterando de acordo com a vida útil do equipamento. De forma geral, os sons emitidos vão se tornando mais agudos ao ouvido humano. Um técnico experiente ao visitar uma subestação é capaz de perceber se o retificador está com algum defeito, ou na iminência de dar problema, apenas pelo ruído emitido.

Baseado nesse conhecimento, um sistema externo ao retificador capaz de classificá-lo em tempo real, de forma preditiva, entre bom, mediano e ruim com boa precisão se apresenta como uma solução mais confiável e segura do que os meios tradicionais de diagnóstico.

Com base nesse conhecimento, o presente trabalho tem por objetivo criar um modelo CNN e treiná-lo por meio de aprendizagem supervisionada com áudios gravados de retificadores em três categorias: bom, mediano e ruim. Após o treinamento, o modelo passa por um processo de otimização para rodar em microcontroladores e, por fim, ele é programado em uma placa Arduino Nano 33 BLE Sense[17], a qual possui uma série de sensores incorporados, dentre eles microfone e LEDs, necessários para aquisição e indicação do estado dos retificadores.

Esse trabalho utilizou como base a biblioteca TensorFlow[25] para Python disponibilizada gratuitamente pelo Google e sua versão para microcontroladores, TensorFlow Lite Micro[16], escrita em C++ e, atualmente, compatível com uma lista limitada de microcontroladores.

2. REVISÃO BIBLIOGRÁFICA

Nessa seção é apresentada de forma sucinta uma revisão sobre os trabalhos relacionados ao emprego de *Machine Learning (ML)* na indústria para diagnóstico de máquinas e equipamentos, bem como a recente utilização de *ML* embarcada em microcontroladores na indústria.

2.1 Uso de Machine Learning na Indústria

Em Ericeira (2019) é apresentado métodos de detecção de defeitos em rolos de transportadores de correia por meio de sensoriamento ultrassônico dos sinais por eles emitidos. O autor utilizou um sensor comercial capaz de captar sons nas frequências de 20 kHz até 100 kHz. O conjunto de dados de áudios de rolos bons e defeituosos foi utilizado para testes de desempenho em dois algoritmos de aprendizagem: *Random Forest* e *Multilayer Perceptron*. Dentre os experimentos realizados pelo autor, o de maior acurácia alcançou média de 83,68% quanto a classificação correta dos rolos, tendo obtido 90% de acerto no melhor resultado dentre todos os testes.

Da Cruz (2020) também propôs um algoritmo baseado em aprendizado de máquinas para detecção de falhas de rolos de transportadores de correia. O trabalho de Da Cruz (2020) se diferenciou do trabalho de Ericeira (2019) principalmente pelo método utilizado para gravação dos ruídos emitidos pelos rolos dos transportadores de correia. Em Da Cruz (2020) o autor utilizou um microfone do tipo direcional convencional com taxa de amostragem de 96 kHz. Dentre os três algoritmos de ML utilizados, o que obteve melhor acurácia foi o KNN (*K-nearest neighbors*) com precisão de 99,1%.

Santos (2020) utilizou uma CNN para identificação de sujeira nas estruturas de transportadores de correia a partir do processamento de imagens RGB obtidas por uma câmera profissional FLIR AX8. Na avaliação do modelo com dados novos do campo a acurácia foi de 89,20% no melhor cenário

Os trabalhos citados não apresentam resultados de validação em campo, apenas resultados de avaliação em laboratório. Além disso, nenhum dos trabalhos apresentados explorou uma solução embarcada que vise a estimativa online de anomalias.

2.2 Uso de ML embarcada em microcontroladores na Indústria

No contexto da Indústria 4.0 a Inteligência Artificial espaço na indústria, seja em análise de grande volume de dados (*big data*) de máquinas conectadas com os sistemas ERP (*Enterprise Resource Planning*), modelo conhecido como *Data-Driven* (WEN, et al, 2018), ou embarcada em instrumentos microprocessados para diagnósticos de máquinas e equipamentos em campo.

Nesse mesmo sentido, Warden et al. (2019) diz que o uso de Inteligência Artificial de forma embarcada em microcontroladores é um campo novo e promissor. Embora, segundo o autor, devido às limitações de recursos dos microcontroladores, como memória RAM na ordem de KB, memória de armazenamento na ordem de MB e *clock* de algumas dezenas de MHz, soluções práticas não eram até então viáveis.

A chegada recente de extensões para microcontroladores de bibliotecas de ML já conhecidas pelo público[15 16] associada ao fato do baixíssimo custo de microcontroladores de 32 bits tem atraído a atenção de desenvolvedoras e entusiastas para novas soluções de ML embarcada para a Indústria (ZHANG et al., 2023).

Figura 1. Arduino Nano 33 BLE Sense. Fonte: (Arduino, 2023)

O Arduino Nano 33 BLE Sense é uma das poucas placas listadas no site do projeto Tensor Flow Lite Micro[16] como compatíveis com essa biblioteca.

3. METODOLOGIA

3.1 Sistemas embarcados e Arduino

Marwedel (2003) define sistema embarcado como um sistema de processamento de informações que são embutidos em produtos, os quais podem ser carros, equipamentos de telecomunicação, maquinários ou mesmo uma placa de circuito microprocessado. Frequentemente sistemas embarcados são conectados ao ambiente físico através de sensores coletando informações sobre o ambiente e atuadores controlando o ambiente. Dentre as várias característica dos sistemas embarcados, as de maior relevância são: que esses sistemas são reativos, pois estão em constante interação com o seu ambiente; são sistemas dedicados a uma determinada aplicação, já que são programados de forma executarem especificamente uma tarefa; são sistemas que precisam ser eficientes, já que rodam em hardware com limitações de recursos.

O Arduino Nano 33 BLE Sense Rev2 é uma placa de baixo custo destinada a pequenas aplicações de IA. A placa conta com uma série de sensores embutidos como sensor de proximidade, sensor de umidade, sensor de pressão, acelerômetro, microfone e um módulo Bluetooth® 5 de baixo consumo. Para sinalização, a placa possuí um LED RGB embutido, o que pode funcionar como um pixel.

As especificações dos chips da placa são singelas. A placa possui um microcontrolador de 64 MHz de clock baseado em Arm® Cortex-M4, 256 KB de memória SRAM e 1 MB de memória Flash para armazenamento do programa e dados do usuário. A tensão interna de operação da placa é de 3.3 Vcc e ela pode executar modelos simples de IA consumindo apenas 17 mA[16]. Devido as limitações do hardware não é possível capturar áudio e treinar o modelo de ML diretamente na placa Arduino.

3.2 TensorFlow Lite Micro

A biblioteca TensorFlow Lite Micro é uma versão reduzida para microcontroladores da versão TensorFlow completa para computadores pessoais. Assim, o treinamento da rede deve ser feito utilizando a infraestrutura de um computador pessoal e depois o modelo é convertido para a versão compatível com a biblioteca para microcontroladores escrita na linguagem C++. Não é possível treinar modelos de AI em microcontroladores utilizando a biblioteca TensorFlow Lite Micro. (TensorFlow, 2023)

Ao pensar em uma solução de IA embarcada, deve-se tomar o cuidado com as operações de pré-processamento de dados, já que essas operações também precisam ser realizadas nos dados aquisitados pelo sistema embarcado. Outro ponto a ser observado é o tamanho final do modelo, já que há limitação de recursos de processamento, memória RAM e memória para armazenamento do modelo (Warden, 2019).

Baseado nessas limitações, foi escolhido o mesmo modelo CNN do exemplo *micro_speech* da própria biblioteca TensorFlow Lite Micro, o qual é adequado em tamanho e eficiência para o objetivo deste trabalho. Trata-se do modelo mostrado de forma gráfica na Figura 2.

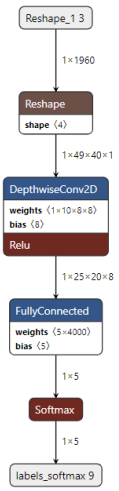

Figura 2. Modelo CNN representado graficamente pelo software Netron[27]. Fonte: autor

O modelo da Figura 2 se assemelha muito ao modelo apresentado na página do projeto Tensor Flow Lite Micro, diferindo apenas na primeira camada, já que para esse projeto houve um processo de quantização do modelo, processo em que a entrada do sistema e os pesos dos neurônios do modelo original, que estavam em formato de ponto flutuante – *floatt16*, foram quantizados em variáveis do tipo inteiro de 8 bits – *int8*. O processo de quantização, quando bem realizado, não implica em redução da acurácia do modelo, podendo reduzir seu tamanho em até 4x e acelerar em até 3x o seu processamento (TensorFlow, 2023).

Todo o treinamento e conversão do modelo foi realizada pelo conjunto de scripts disponibilizados para o exemplo *micro_speech* da biblioteca TensorFlow Micro Lite substituindo o conjunto de dados original do exemplo pelos arquivos de áudios das gravações dos retificadores armazenados em pastas com o nome de suas classes: bom, mediano e ruim.

3.3 Construção da base de dados do modelo

Para esse trabalho foram feitas gravações dos sons de retificadores trifásicos em operação nas subestações de distribuição da empresa CEMIG. Foram considerados 15 equipamentos, o quais foram classificados da seguinte maneira: 10 em bom estado de funcionamento, 3 em estado

mediano (possível falha iminente) e 2 em estado defeituoso. A classificação foi realizada com o auxílio do técnico especialista, o qual utilizou instrumentos como osciloscópio e analisador de energia para verificação da presença de componentes harmônicas na saída do retificador. Com base nessas informações, foram definidas três classes para os retificadores: bom, mediano e ruim.

As gravações foram feitas por meio de aparelho do aparelho celular Google® Pixel 4 XL com uso de aplicativo específico para gravação de áudio configurado para gravar em formato *wave* puro (.wav) com uma taxa de amostragem de 44100 Hz e 32 bits de resolução, conforme mostrado na Figura 2. Foram gravados áudios de 10 minutos para cada retificador.

Figura 2. Processo de gravação dos áudios dos retificadores. Fonte: autor

Cada áudio de 10 minutos foi separado em trechos de 1 segundo, produzindo 600 amostras para cada retificador. Devido à baixa disponibilidade de equipamentos com defeito em campo, a quantidade de amostras da classe ruim ficou limitada a 1200 amostras. Para efeito de padronização, foram utilizadas 1200 amostras de cada uma das classes tomadas de forma aleatória. Contudo, as amostras de 4 retificadores da classe "bom" e de 1 retificador da classe "mediano" não foram utilizadas durante as etapas de treinamento e validação, sendo reservadas para a etapa de teste do sistema. Das 1200 amostras de cada classe, 1000 foram utilizadas para treinamento da rede e 200 para validação do algoritmo.

Apesar da possibilidade de trabalhar com modelos de CNN com dados de áudio puro, trabalhos recentes, bem como a natureza dos sons dos retificadores, mostram que com o pré-processamento dos áudios no domínio da frequência a rede é capaz de obter melhores resultados[21] [22] [23].

O pré-processamento dos dados, treinamento do modelo, conversão, quantização e otimização para microcontroladores foi realizado utilizando a linguagem Python e as bibliotecas Librosa[26] e TensorFlow[25].

O modelo adaptado para microcontroladores foi programado e testado na placa Arduino Nano 33 BLE Sense Rev2 utilizando a biblioteca Tensor Flow Lite Micro para Arduino.

Abaixo são descritas as etapas necessárias para o treinamento do modelo, desde o pré-processamento dos áudios até a implementação do modelo no microcontrolador.

4. DESENVOLVIMENTO

4.1 Pré-processamento dos áudios

O mesmo pré-processamento usado no treinamento do modelo deve ser realizado no momento da inferência no sistema embarcado, já que a rede neural precisa ser alimentada exatamente com o mesmo tipo de dado pela qual foi treinada para que ela consiga classificar a entrada baseada no conhecimento adquirido no treinamento.

Todos os áudios passaram pelo processo de *resampling* saindo de 44100 Hz para 16000 Hz. Isso se fez necessário devido a uma limitação o hardware do Arduino em que a taxa máxima de amostragem é de 16 kHz. Como mostrado na Figura 4, a maior diferença entre as três classes de classificação no domínio da frequência está em frequências superiores a 8 kHz.

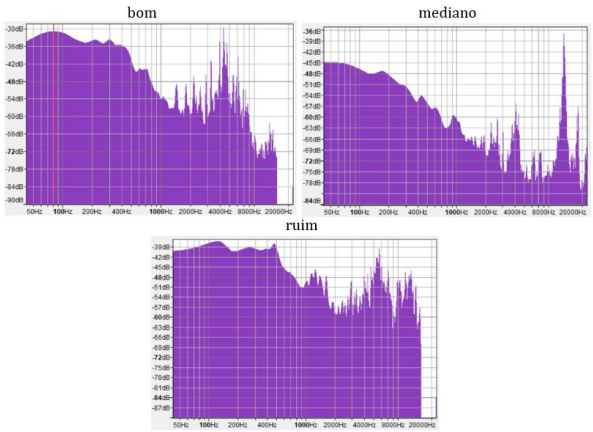

Figura 4. Espectro de frequência de áudio das três classes. Fonte: autor

Pelo Teorema da Amostragem de Nyquist-Shannon (Oppenheim, 2012), sinais amostrados na frequência de 16 kHz só guardam informações de frequências até 8 kHz. Isto é, a limitação do hardware embarcado exige que a parte mais significativa, do ponto de vista de classificação, seja perdida. Apesar disso, o modelo ainda apresentou boa acurácia em testes.

Após esse processo, os áudios passaram pelo algoritmo de STFT (do inglês: *Short-Time Fourier Transform*) com uma janela de 30 ms e saltos de 20ms para a construção do espectrograma. Cada janela de 30ms produz 240 informações de frequência, e cada conjunto de seis informações foi tomada a média. Isso produziu 40 informações de frequência por janela. A janela foi se movendo com salto de 20ms fazendo com que esse processo se repetisse 49 vezes, conforme mostrado na Figura 5. O resultado foi uma matriz de tamanho 49x40 com o total de 1960 informações. Essa é a quantidade de neurônios da primeira camada do modelo.

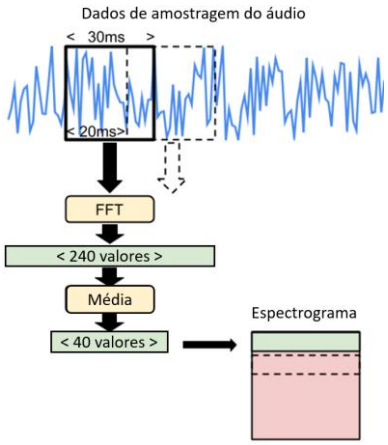

Figura 5. Representação gráfica do pré-processamento. Fonte: Micro Speech Training[16] adaptado pelo autor

A matriz gerada por esse processo pode ser plotada como um espectrograma, conforme mostrado na Figura 6. Assim, todas as técnicas utilizadas em modelos de visão computacional podem ser utilizadas a partir desse ponto. É o que ocorre na camada do meio da rede. Já a última camada possui uma função de ativação e apenas cinco neurônios, que é exatamente o número de classes possíveis no modelo.

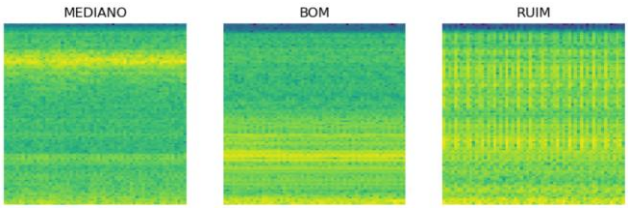

Figura 6. Espectrograma de exemplos das três classes utilizadas. Fonte: autor

4.1 Treinamento do modelo

O script em Python disponível para treinamento utiliza um *dataset* de áudios de gravação de sons ambientes da biblioteca *speech_micro* para mixar com os áudios originais dos retificadores na proporção de 10%. Isso ajuda a evitar o *overfitting* e garante a generalização do modelo e bom funcionamento mesmo com som ambiente.

O script também cria duas classes para classificação: *silence* e *unknown*. Isso é muito importante do ponto de vista do sistema embarcado, pois o torna capaz de sinalizar através do LED quando um som não é reconhecido ou há silêncio. Com isso o modelo passa a ter cinco classes: *silence*, *unknown*, bom, mediano, ruim.

4.3 Conversão e utilização do modelo

Com a utilização de *scripts* em Python foi feita a conversão do modelo para a biblioteca TensorFlow Lite Micro. Essa conversão consiste, basicamente, em pegar o modelo treinado com os respectivos pesos de cada neurônio e transformá-lo em um arquivo .cpp cujo conteúdo é a definição de um grande vetor de bytes contendo os parâmetros treinados da CNN.

O arquivo .cpp gerado foi importado no projeto da biblioteca TensorFlow Lite Micro. Após compilação e envio para o Arduino a predição já se torna operacional. Foram feitos pequenos ajustes no código fonte disponibilizado pela biblioteca TensorFlow Lite Micro a fim de personalizar as classes e as cores do LED pelas quais cada classe é identificada.

5. RESULTADOS

Após o processo de treinamento foi executada uma etapa de validação com áudios não utilizados durante o treinamento, obtendo uma acurácia de 97,8%, gerando a matriz de confusão da Tabela 1. Contudo, testes realizados reproduzindo os áudios do subconjunto de testes para inferência direto no sistema embarcado alcançou uma acurácia de aproximadamente 68%, o que, de certa forma, já era esperado devido ao hardware da placa não ser adequado para aplicações comerciais e ser propenso a falhas, como perda de amostras durante o sensoriamento.

Tabela 1 – Matriz de Confusão

CLASSE		PREVISTO				
		Silence	Unknown	Bom	Mediano	Ruim
REAL	Silence	185	0	9	5	1
	Unknown	0	0	0	0	0
	Bom	3	0	196	1	0
	Mediano	0	0	0	200	0
	Ruim	0	0	0	0	200

Os áudios gravados a taxa de 44100 Hz possuem mais informações de frequência em relação àqueles que foram utilizados no modelo embarcado. O *dataset* de áudios amostrados a 44100 Hz foi utilizado em uma CNN mais robusta tendo obtido taxa de acurácia de 100% após apenas 3 épocas de treinamento.

Acredita-se que com a utilização de áudio de melhor qualidade em placas mais especializadas, tal como a Arduino Portenta Vision ou a Arduino Portenta H7, será possível obter resultados tão bons quanto os experimentados no computador.

6. CONSIDERAÇÕES FINAIS

Neste trabalho foi apresentada uma solução embarcada de baixo custo para classificação em tempo real de retificadores trifásicos entre bom, mediano e ruim utilizando bibliotecas *open source* para treinamento e implementação na placa Arduino Nano 33 BLE Sense Rev2.

O uso de IA para diagnóstico de falhas em equipamentos da indústria é uma área relativamente nova e promissora (Jagatheesaperumal, 2021). No setor de energia elétrica a presença de IA ainda é insignificante, apesar de existir um potencial enorme para sua aplicação.

6.1 Sugestões para trabalhos futuros

Como sugestões de trabalhos futuros pode-se explorar se o processo de quantização teve influência na queda da precisão do modelo embarcado e qual é o ganho em performance na utilização dessa técnica, além de avaliar o consumo do modelo original frente ao modelo quantizado.

É sugerido também a utilização do mesmo framework para treinamento de modelos com utilização de base de dados de outros equipamentos, bem como utilização de outras sinais como vibração e temperaturas.

Por último, pode-se fazer um estudo comparativo entre os diversos frameworks e algoritmos de ML disponíveis para sistema embarcados.

AGRADECIMENTOS

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior, Brasil (CAPES), Código de Financiamento 001; do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq); do Instituto Tecnológico Vale (ITV); da Universidade Federal de Ouro Preto (UFOP); da Fundação de Amparo à Pesquisa do Estado de Minas Gerais (FAPEMIG); e da Vale S.A. G. Pessin agradece o apoio do CNPQ, projeto 306423/2020-0.

REFERÊNCIAS

1. Barchiesi, D., Giannoulis, D., Stowell, D., & Plumbley, M. D. (2015). *Acoustic scene classification: Classifying environments from the sounds they produce. IEEE Signal Processing Magazine*, 32(3), 16-34.
2. Zhu, B., Xu, K., Wang, D., Zhang, L., Li, B., & Peng, Y. (2018). *Environmental sound classification based on multi-temporal resolution convolutional neural network combining with multi-level features. In Advances in Multimedia Information Processing–PCM 2018: 19th Pacific-Rim Conference on Multimedia, Hefei, China, September 21-22, 2018, Proceedings, Part II 19 (pp. 528-537). Springer International Publishing*.
3. Rettberg, A., Zanella, M., Domer, R., Gerstlauer, A., & Rammig, F. (Eds.). (2010). *Embedded System Design: Topics, Techniques and Trends: IFIP TC10 Working Conference: International Embedded Systems Symposium* (IESS), May 30-June 1, 2007, Irvine (CA), USA (Vol. 231). Springer.
4. Barbosa, L. C. M. (2012). Áudio Digital: Uma Abordagem ao Audio pela Perspectiva de Ensino (Doctoral dissertation, Instituto Politecnico do Porto (Portugal)).
5. Peter M. (2003) *Embedded Systems Design. Kluwer Academic Publishers*.
6. De Oliveira, S. (2017). Internet das Coisas com ESP8266, Arduino e Raspberry Pi. Novatec Editora Ltda.
7. Platts, J., ST. Aubyn, J. (1992). *Uninterruptible Power Supplies. 1. ed. London: Peregrinus on behalf of the Institution of Electrical Engineers*, 130 p.
8. Emadi, A., Nasiri, A., Bekiarov, B. S. (2015). *Uninterruptible Power Supplies and Active Filters*. 1. ed. *New York: CRC Press,* 285p.
9. Silva, I. N.; Spatti, D. H.; Flauzino, R. A. (2010) Redes neurais artificiais para engenharia e ciências aplicadas.
10. Audacity Software. “Audacity” (2023). https://www.audacityteam.org/
11. Francois, C. (2017). *Deep learning with python – Second Edition. Manning Publications. New York, NY*.
12. Laurence, M. (2020). *AI and Machine Learning for Coders: A Programmer's Guide to Artificial Intelligence. O'Reilly Media, Inc.*
13. Géron, A. (2022). *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow. " O'Reilly Media, Inc."*.
14. Warden, P., & Situnayake, D. (2019). *Tinyml: Machine learning with tensorflow lite on arduino and ultra-low-power microcontrollers*. O'Reilly Media.
15. *MicroMLGen library for Python | Eloquent Arduino. (2022, December.). https://eloquentarduino.com/libraries/micromlgen/*
16. *TensorFlow Lite for Microcontrollers. (2023, March). TensorFlow. Retrieved May 29, 2023, from https://www.tensorflow.org/lite/microcontrollers*
17. *Nano 33 BLE Sense Rev2. (2022, November). Arduino. Retrieved May 29, 2023, from https://docs.arduino.cc/hardware/nano-33-ble-sense-rev2*
18. Zhang, Z., & Li, J. (2023). *A Review of Artificial Intelligence in Embedded Systems. Micromachines*, 14(5), 897.
19. Jagatheesaperumal, S. K., Rahouti, M., Ahmad, K., Al-Fuqaha, A., & Guizani, M. (2021). *The duo of artificial intelligence and big data for industry 4.0: Applications, techniques, challenges, and future research directions. IEEE Internet of Things Journal, 9(15), 12861-12885*.
20. Carvalho Júnior, J. R. D. (2018). Processamento digital de imagens para a identificação automática de falhas em rolos dos transportadores de correias. [Dissertação de Mestrado, UFOP]. https://www.repositorio.ufop.br/handle/123456789/10070
21. Cruz, E. B. D. (2020). Detector de falhas de rolos de transportadores de correia baseado em aprendizado de máquina. [Dissertação de Mestrado, UFOP]. https://repositorio.ufop.br/handle/123456789/13038
22. Ericeira, D. R. (2019). Detecção automática de defeitos em rolos de transportadores de correia utilizando sensoriamento ultrassônico. [Dissertação de Mestrado, UFOP]. https://www.repositorio.ufop.br/handle/123456789/12554
23. Silva, A. C. P. (2021). Monitoramento da qualidade de SINTER FEED através de dados espectrais associados a aprendizado de máquina–estudo de caso: Mina de Carajás Serra Sul (S11D).
24. Santos, A. A. (2020). Sistema automático para a inspeção visual de transportadores de correia por meio de redes neurais convolucionais. [Dissertação de Mestrado, UFOP] https://www.repositorio.ufop.br/handle/123456789/13097
25. *TensorFlow. (2023, March). TensorFlow. Retrieved May 29, 2023, from https://www.tensorflow.org/*
26. *Librosa. (2023, March). Retrieved May 29, 2023, from https://librosa.org/doc/latest/index.html*
27. *Netron. (2023, May).). Retrieved May 29, 2023, from https://github.com/lutzroeder/netron*
28. *Oppenheim, A. V., & Schafer, R. W. (2012). Processamento em tempo discreto de sinais. Tradução Daniel Vieira. 3ª ed.-São Paulo: Pearson Education do Brasil.*
29. *Valerio Velardo. (n.d.). Valerio Velardo. Retrieved June 7, 2023, from https://valeriovelardo.com/*