DOI: 10.48011/asba.v2i1.1269

Aplicação de técnicas de *machine learning* para o monitoramento de vibração e detecção de trincas em pontes

Wander P. Jesus [*] **Evandro J. G. Lima** [**] **Rafael G. Alvares** [***]
Joyce R. Silva [****] **Douglas A. Rocha** [†] **Zélia M. A. Peixoto** [‡]

[*] *Faculdade de Engenharia Eletrônica e de Telecomunicação, Pontifícia Universidade Católica de Minas Gerais, MG, (e-mail: wander.pjesus@gmail.com).*
[**] *Faculdade de Engenharia Eletrônica e de Telecomunicação, Pontifícia Universidade Católica de Minas Gerais, MG, (e-mail: evandrojgl@gmail.com).*
[***] *Faculdade de Engenharia Eletrônica e de Telecomunicação, Pontifícia Universidade Católica de Minas Gerais, MG, (e-mail: rafalvares@gmail.com).*
[****] *Faculdade de Engenharia Eletrônica e de Telecomunicação, Pontifícia Universidade Católica de Minas Gerais, MG, (e-mail: joyce.eng07@gmail.com).*
[†] *Faculdade de Engenharia Eletrônica e de Telecomunicação, Pontifícia Universidade Católica de Minas Gerais, MG, (e-mail: douglasabreudarocha@gmail.com).*
[‡] *Faculdade de Engenharia Eletrônica e de Telecomunicação, Pontifícia Universidade Católica de Minas Gerais, MG, (e-mail: assiszmp@pucminas.br).*

Abstract: Excessive vibrations and cracks are some of the main causes of anomalies in civil structures such as bridges and viaducts, requiring continuous monitoring and evaluation in order to enable preventive actions. In this context, this work proposes machine learning techniques for structural health monitoring systems (SHM). The methodology adopted includes a vibration monitoring system and bridge crack detection. The classification of the vibration signals is performed in four categories (no vibration, normal, alert and critical) through a decision tree (DT), and the detection of cracks through simulation, to classify the structure in relation to the presence or absence of cracks, using a convolutional neural network (CNN).

Resumo: Vibrações excessivas e trincas são algumas das principais causas de anomalias em estruturas civis como pontes e viadutos, demandando acompanhamento e avaliações contínuas com o propósito de viabilizar ações preventivas. Nesse contexto, este trabalho propõe técnicas de *machine learning* para sistemas de monitoramento de saúde estrutural (SHM). A metodologia adotada engloba um sistema de monitoramento de vibração e detecção de trincas em pontes. A classificação dos sinais de vibração é executada em quatro categorias (sem vibração, normal, em alerta e crítico) através de uma árvore de decisão (DT), e a detecção de trincas através de simulação, para classificar a estrutura em relação à presença ou ausência de trincas, utilizando uma rede neural convolucional (CNN).

Keywords: Vibration, Cracks, Structural Health Monitoring, Bridges, Convolutional Neural Network, Decision Tree.

Palavras-chaves: Vibração, Trincas, Monitoramento de Saúde Estrutural, Pontes, Rede Neural Convolucional, Árvore de Decisão.

1. INTRODUÇÃO

As pontes e os viadutos são estruturas civis fundamentais no cenário atual devido à sua importância para o trânsito e conexão entre locais. Em consequência da ampla utilização, é fundamental que essas estruturas sejam confiáveis e seguras para os usuários, evitando-se danos patrimoniais e acidentes graves como perdas de vidas humanas.

As vibrações são naturais em estruturas civis visto que estão sujeitas constantemente a efeitos mecânicos. Os problemas ocorrem quando as variações de rigidez e massa são significativas para que a intensidade das vibrações se torne elevada, Subramanian (2013). As estruturas podem vibrar em amplitudes cada vez maiores e, sob o efeito de ressonância, ocasionar rupturas irreversíveis, Serway et al. (2018).

No processo de cura do concreto usado na construção de pontes, as trincas são praticamente inevitáveis devido à contração da estrutura durante a secagem e umedecimento do material. Entretanto, trincas causadas por carga cíclica podem ser graves e propagar ao longo da superfície, acarretando perda de resistência, variações na rigidez e na flexibilidade da estrutura. Os processos naturais também são causadores de anomalia em pontes e viadutos. Exposição ao calor e chuvas excessivas podem, em muitos casos, degenerar e corroer toda a estrutura metálica, além de contribuir para o surgimento das trincas, Subramanian (2013).

A maioria das pontes e viadutos são construções antigas e requerem monitoramento constante da sua saúde estrutural. Os métodos de inspeção visual são poucos confiáveis, exigem tempo e muitas vezes podem ser ineficientes pois envolvem a intervenção humana, Abdulkarem et al. (2019). Algumas técnicas não destrutivas são propostas para medição de vibrações e detecção de trincas em pontes. As mais comuns envolvem aplicações de Sistemas Microeletromecânicos (MEMS), Jesus et al. (2019), e sensores tais como os piezoelétricos, infravermelhos, ultrassônicos e fibras ópticas. Em se tratando da infraestrutura necessária para coletas das informações dos sensores, podem ser utilizados sistemas cabeados ou também as Redes de Sensores sem Fio (WSN), os quais vêm se destacando em estudos recentes. Além da medição de vibração, alguns sistemas são usados também para detecção de trincas, Abdulkarem et al. (2019).

Fernandez et al. (2017) apresentaram um sistema de detecção e classificação de trincas em asfalto utilizando uma DT. Árvore de decisão foi gerada pelo algoritmo J48 e treinada previamente através de um *dataset* contendo 600 imagens de imagens de asfalto ,com trincas e sem trincas . Os autores relataram a utilização da técnica de validação cruzada no processo de treinamento.

Avci et al. (2018) apresentaram uma CNN 1D juntamente com uma WSN para medição de vibração. O método de detecção de danos proposto operou diretamente nos sinais brutos da condição de vibração do ambiente, sem filtragem ou pré-processamento. O *dataset* gerado foi coletado diretamente na estrutura simulando situação com danos e sem danos. Um algoritmo do MATLAB® foi usado para agrupar os sinais, dividi-los em quadros e normalizá-los. 85% dos quadros foram usados para treinamento da rede e 15% para validação.

1.1 Árvore de decisão

Árvore de decisão é um recurso de *machine learning* que classifica dados através da divisão do conjunto de domínios do problema. A técnica consiste em uma árvore binária em que cada nó é uma estrutura do tipo *if-then-else* contendo um teste sobre algum atributo, Quinlan (2014). O algoritmo de aprendizado utiliza amostras de casos, nos quais se conhecem as classificações verdadeiras, para aplicar condições pré-definidas. Para diferenciar as categorias, um resultado esperado deve ser designado para cada padrão de dados.

Para a implementação do classificador existem algoritmos para gerar e treinar as DTs, como em Sharma et al. (2014) que testaram alguns classificadores para averiguar qual o mais indicado para detecção de falhas de ignição de motores através de sinais de vibração. O algoritmo classificador J48 é utilizado para gerar uma DT, e em Gunasegarana et al. (2019), ele foi empregado para diagnóstico de falhas de sistemas de engrenagens retas e, segundo os autores, esse modelo de decisão é útil na seleção dos melhores recursos necessários para a classificação, onde o melhor recurso é posicionado no topo da árvore, seguido pelo segundo recurso mais relevante e, assim, sucessivamente. A Figura 1 mostra o diagrama em blocos de uma árvore de decisão genérica. No bloco raiz é iniciada a classificação contendo o primeiro teste, os próximos nós contêm as regras, que recebem um sinal de entrada e determinam as subdivisões de acordo com as condições pré-definidas. Os ramos são o conjunto de nós até as folhas, que definem os possíveis resultados dos nós.

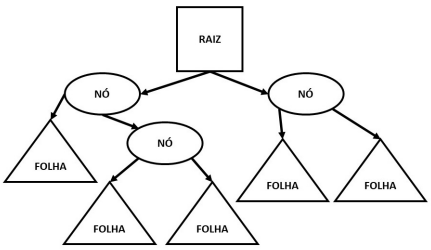

Figura 1. Modelo genérico de uma DT.

1.2 Rede Neural Convolucional

CNN é uma ferramenta de visão computacional e um tipo de modelo de aprendizado profundo que ganhou destaque e popularidade a partir dos resultados obtidos na Competição ImageNet (ILSVRC - *Large Scale Visual Recognition Competition*) em 2012, Krizhevsky et al. (2017). Na indústria, a CNN pode ser aplicada para a inspeção visual autônoma com intuito de otimizar processos de fabricação, Weimer D et al. (2016). Na medicina, os exames de imagens são comuns para diagnósticos de doenças. Li et al. (2014) usaram uma CNN para aprender de forma automática e eficiente os detalhes intrínsecos de imagens pulmonares para classificar doenças. Em sistemas de Monitoramento de Saúde Estrutural (SHM), as CNNs são

usadas para resolução de problemas de danos estruturais em pontes, tais como as trincas, Wei et al. (2019).

A arquitetura de uma CNN é composta principalmente por uma camada de entrada, geralmente sinais relacionados a uma imagem, como por exemplo, os canais RGB, as múltiplas camadas de convolução seguidas por camadas densamente conectadas e a camada de saída. A camada de entrada apenas repassa os sinais de entrada para a camada seguinte. As camadas de convolução realizam a extração de *features*, começando com os de baixo nível, como por exemplo linhas, *loops*, bordas e formas mais primitivas, até os de alto nível, tais como os olhos, bocas e as formas mais complexas, Aggarwal (2018). Essas *features* são extraídas através de *kernels*, que são conjuntos de pesos que ligam as entradas de um neurônio às saídas dos neurônios da camada anterior. Uma camada pode conter vários *kernels* sendo cada um correspondente a uma *feature*. As Camadas Totalmente Conectadas (FCLs) da CNN classificam a imagem, gerando o resultado final. Tal arquitetura é eficiente em aplicações de visão computacional, permitindo treinamento e execução com baixo custo computacional, Aron et al. (2016).

As CNNs caracterizam-se por suas camadas de convolução, em que cada neurônio se conecta apenas aos neurônios de uma pequena região da camada anterior. Essas camadas possibilitam a extração automática de *features*. Assim, a CNN pode ser compreendida como uma modificação na rede Perceptron Multicamadas (MLP), inserindo camadas convolucionais entre a camada de entrada e as FLCs. Na Figura 2, a camada de entrada faz a leitura de uma imagem, as camadas de convolução fazem a extração de *features*, as camadas totalmente conectadas classificam a imagem e a camada de saída indica a classe a qual a imagem pertence.

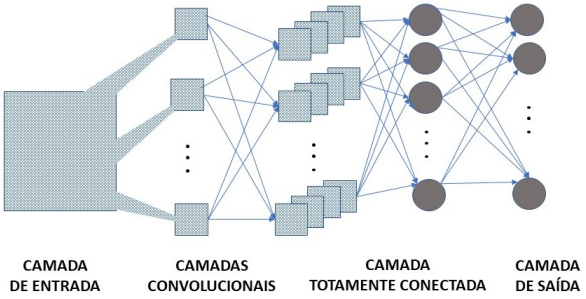

Figura 2. Arquitetura de uma CNN bidimensional.

2. SISTEMA DE MONITORAMENTO DE VIBRAÇÃO E DETECÇÃO DE TRINCAS

Este trabalho apresenta a implementação de um sistema para monitoramento de vibração e a simulação de detecção de trincas em pontes com o intuito de avaliar e classificar a saúde estrutural dessas construções. A classificação dos sinais de vibração são feitas em quatro classes (sem vibração, normal, em alerta e crítico) através de uma árvore de decisão (DT), e a detecção de trincas através de simulação, na qual uma rede neural convolucional (CNN) classifica a estrutura sob monitoramento em relação a presença ou ausência de trincas. Ambas as técnicas de classificação são consolidadas por aprendizado supervisionado, no qual os classificadores foram treinados utilizando *datasets* com dados rotulados para cada classe, gerados pelo próprio sistema, a partir de testes realizados em uma estrutura de escala reduzida. O sistema engloba, portanto, aplicação de dois métodos de *machine learning*. A implementação do *hardware* do sistema utiliza dois kits de desenvolvimento da STMicroelectronics®. Os dados são apresentados na plataforma de interface gráfica Unicleo®, por meio de um computador, e pelo aplicativo ST BLE Sensor®, por meio de um *smartphone* via comunicação *bluetooth*. Os resultados são disponibilizados aos usuários, para visualização e análise, em uma *dashboard*. A Figura 3 mostra o diagrama em blocos do sistema de monitoramento de vibração e detecção de trincas proposto. Em (a), a simulação é consolidada por imagens da estrutura classificadas por uma CNN, com ou sem trincas. Em (b), é implementado o sistema de monitoramento de vibração por meio do SensorTile.Box, responsável pela classificação da vibração, por meio de uma DT e por apresentar os dados em um computador.

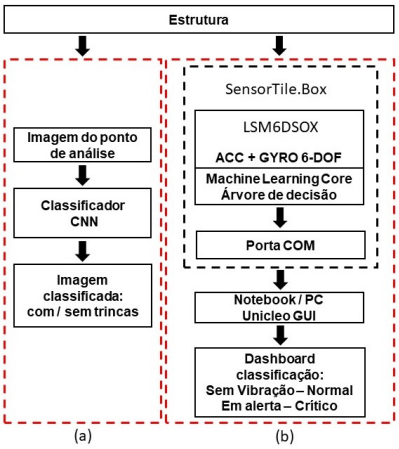

Figura 3. Sistema de medição: Detecção de trincas e monitoramento de vibração

2.1 Hardware

O SensorTile.Box é um kit de desenvolvimento da ST Microelectronics® para aplicações em internet das coisas (IoT) com conexão *bluetooth*. O *hardware* possui um microcontrolador STM32L4R9ZIJ6® *ultra low power*, com núcleo ARM® cortex, Processador de Sinal Digital (DSP) e Unidade de Ponto Flutuante (FPU). O módulo LSM6DSOX, presente no kit, possui um acelerômetro e um giroscópio embutidos, abrangendo uma faixa de aceleração com fundos de escala de 2, 4, 8 e 16 unidades de aceleração, sensibilidade de 0.061 (mg/bit) e uma faixa de taxa angular de 125, 250, 500, 1000, 1000 e 2000 graus por segundos (DPS) com sensibilidade de 4.375 (mdps/bit). O módulo também possui embutido um núcleo de *machine learning* (MLC) que opera com padrões de dados provenientes dos sensores. As informações de entrada podem ser filtradas através de um bloco dedicado programável contendo um filtro de resposta infinita ao impulso (IIR). O processamento lógico pode ser obtido simultaneamente por até 8 árvores de decisão, com até 16 resultados, composta por uma série de nós configuráveis, caracterizados por comandos condicionais *if-then-else* em

que os valores das (*features*) são avaliadas em relação aos limites definidos, STMicroelectronics (2019). Neste trabalho, o módulo LSM6DSOX é responsável por medir as vibrações da estrutura através do sensor acelerômetro.

3. PROCESSO DE MONITORAMENTO DE VIBRAÇÃO

Técnicas de processamento de sinais de vibração podem ser utilizadas para avaliar mudanças em parâmetros estruturais de pontes. As respostas dinâmicas das estruturas são convertidas em grandezas digitais por dispositivos inteligentes capazes de captar oscilações avaliadas em ambiente computacional com a finalidade de determinar padrões, Beheshti et al. (2020). Os impactos das vibrações podem ser avaliados segundo regulamentações quanto aos níveis que podem ou não apresentar danos às estruturas, Bachmann et al. (1994). Dentre os métodos disponíveis para classificação dos sinais de vibração, optou-se pela DT, visando explorar a capacidade de implementá-la no kit de desenvolvimento SensorTile.Box.

3.1 Treinamento, testes e resultados

As frequências em pontes geralmente são dependentes do comprimento do vão e do tipo de construção. Embora possuam faixas de frequência entre 2 (Hz) e 4 (Hz), podem variar em uma faixa de frequência de 0 a 14 (Hz), Bachmann et al. (1994). Os testes de monitoramento e classificação de vibração neste trabalho, foram executados em uma estrutura de madeira, em escala reduzida, com intuito de validar o funcionamento do *hardware* e a técnica de classificação de vibração por uma DT. A vibração mecânica do sistema foi executada por um *shaker*, excitado por um gerador de sinais configurado em 60 (Hz). Um *firmware* intermediário foi desenvolvido no *software* Algobuilder® para a geração do *dataset* pelo próprio acelerômetro embutido no dispositivo LSM6DSOX. Os dados para o *dataset* foram coletados pelo *software* Unicleo® e salvos em planilhas com extensão csv. Nessa etapa, os dados foram coletados para indicar o estado normal, em alerta e crítico, através dos sinais de vibração. Esses estados foram gerados a partir da remoção de algumas peças da estrutura e registrados por meio das variações nos valores das *features* (Pico a pico, máximo, mínimo e passagem por zero). Cada *dataset* contou com aproximadamente 10.000 amostras de sinais do acelerômetro nos três eixos, coletados a uma taxa de saída de dados (ODR) de 104 (Hz).

Na etapa de treinamento, foi utilizado o software Unico® para gerar e treinar a DT a partir do *dataset* coletado, e para configurar o filtro passa-faixa e a ODR do MLC. Na etapa final, o código gerado no treinamento para a DT, foi programado no MLC a partir de um novo *firmware* gerado pelo Algobuilder®, onde foram adicionados novos recursos, como por exemplo, o cálculo das transformadas rápidas de Fourier (FTTs). A presença do MLC configura o sistema como uma aplicação de *Edge computing*, onde o processamento dos sinais e a classificação pela DT, são executados no próprio sensor, reduzindo a necessidade de plataformas externas e evitando a programação deste processamento no microcontrolador principal do SensorTile.Box. Isto possibilita aplicações com utilização de ML em *hardware*, com baixíssimo consumo de energia. A Figura 4 mostra o diagrama em blocos de configuração do *machine learning core* do LSM6DSOX. Os sinais de vibração são coletados pelo acelerômetro em três eixos. Em seguida, o filtro IIR foi configurado como passa-faixa. As *feature* utilizadas para treinar a DT foram os valores de pico a pico, máximo e mínimo (em g) e quantidade de passagem por zero. Após a classificação da DT, para evitar a ocorrência de falsos positivos e exibir resultados com menor probabilidade de erro, foram adicionados filtros na saída da DT, chamados de meta-classificadores. Estes meta-classificadores consistem em janelas de verificação dos resultados da árvore de decisão. Neste trabalho foram adicionados filtros de uma janela para o estado em alerta e duas janelas para o estado crítico. Isto significa que, ao verificar uma saída em estado crítico da árvore de decisão, por exemplo, o MLC aguarda ainda a ocorrência de dois resultados em estado crítico consecutivos para enviar este novo estado para a saída.

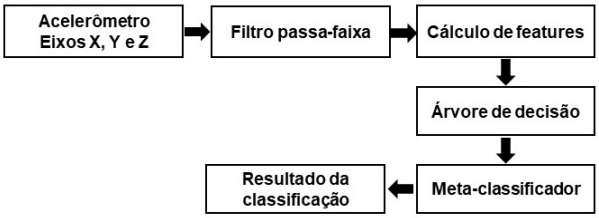

Figura 4. Diagrama em blocos: Machine learning core.

O *shaker* foi utilizado para vibrar mecanicamente a estrutura, tanto na etapa de treinamento da DT quanto na etapa de classificação. A ODR do MLC foi configurada em 104 (Hz) e o filtro passa-faixa foi configurado com uma frequência de corte superior de 50 (Hz), adequando o sinal obtido ao critério de Nyquist, e a frequência de corte inferior de 0.8 (Hz), para remover a componente contínua do sinal. O comprimento da janela de amostragem para cálculo das *features* foi configurado para 104 amostras, criando assim, um intervalo de 1 segundo entre cada cálculo. Na Figura 5, a árvore de decisão foi gerada e treinada utilizando o algoritmo J48, pelo software Unico®. Na etapa de treinamento foi utilizada a técnica de validação cruzada 10-fold com intuito de generalizar o *dataset*.

As métricas do treinamento são mostradas na Tabela 1 e na Figura 6.

A matriz de confusão para as instâncias classificadas na DT gerada em relação aos estados sem vibração (*novib*), normal (*normal*), em alerta (*alert*) e crítico (*critical*). A diagonal principal mostra as instâncias classificadas corretamente e as demais posições da matriz mostram os erros de classificação.

A Tabela 1 exibe o número de instâncias (janelas de cálculos das *features*) total, classificadas correta e incorretamente. A acurácia da DT obtida nesse treinamento foi de 99.1667%, para as instâncias analisadas.

No primeiro teste, a estrutura foi mantida íntegra e a classificação indicou estado normal de vibração. Em seguida, os parafusos de sustentação da peça central entre as bases da estrutura foram removidos, e a classificação indicou estado de alerta e, finalmente, os parafusos de sustentação de uma das bases foram removidos, e o sistema

```
F2_PeakToPeak_on_ACC_V2 <= 0.0292969
|   F1_PeakToPeak_on_ACC_V <= 0.00390625: novib (90.0)
|   F1_PeakToPeak_on_ACC_V > 0.00390625: critical (89.0)
F2_PeakToPeak_on_ACC_V2 > 0.0292969
|   F9_ZeroCross_on_filter_BP_on_ACC_V2 <= 45
|   |   F18_MAX_on_ACC_V <= 1.01465
|   |   |   F20_MAX_on_filter_BP_on_ACC_V <= 0.00720215
|   |   |   |   F3_PeakToPeak_on_filter_BP_on_ACC_V <= 0.0153046: normal (1.0)
|   |   |   |   F3_PeakToPeak_on_filter_BP_on_ACC_V > 0.0153046: alert (11.0)
|   |   |   F20_MAX_on_filter_BP_on_ACC_V > 0.00720215
|   |   |   |   F9_ZeroCross_on_filter_BP_on_ACC_V2 <= 38
|   |   |   |   |   F7_ZeroCross_on_ACC_V2 <= 35
|   |   |   |   |   |   F16_MIN_on_filter_BP_on_ACC_V <= -0.00810242
|   |   |   |   |   |   |   F4_PeakToPeak_on_filter_BP_on_ACC_V2 <= 0.0351257: alert (6.0)
|   |   |   |   |   |   |   F4_PeakToPeak_on_filter_BP_on_ACC_V2 > 0.0351257
|   |   |   |   |   |   |   |   F1_PeakToPeak_on_ACC_V <= 0.015625: normal (2.0)
|   |   |   |   |   |   |   |   F1_PeakToPeak_on_ACC_V > 0.015625: alert (1.0)
|   |   |   |   |   |   F16_MIN_on_filter_BP_on_ACC_V > -0.00810242
|   |   |   |   |   |   |   F9_ZeroCross_on_filter_BP_on_ACC_V2 <= 9: critical (1.0)
|   |   |   |   |   |   |   F9_ZeroCross_on_filter_BP_on_ACC_V2 > 9: normal (2.0)
|   |   |   |   |   F7_ZeroCross_on_ACC_V2 > 35: normal (5.0)
|   |   |   |   F9_ZeroCross_on_filter_BP_on_ACC_V2 > 38: normal (12.0)
|   |   F18_MAX_on_ACC_V > 1.01465: alert (64.0/2.0)
|   F9_ZeroCross_on_filter_BP_on_ACC_V2 > 45
|   |   F4_PeakToPeak_on_filter_BP_on_ACC_V2 <= 0.0360107: normal (63.0/1.0)
|   |   F4_PeakToPeak_on_filter_BP_on_ACC_V2 > 0.0360107
|   |   |   F3_PeakToPeak_on_filter_BP_on_ACC_V <= 0.0189056: alert (6.0)
|   |   |   F3_PeakToPeak_on_filter_BP_on_ACC_V > 0.0189056
|   |   |   |   F6_ZeroCross_on_ACC_V <= 6: normal (3.0)
|   |   |   |   F6_ZeroCross_on_ACC_V > 6: alert (1.0)

Number of Leaves  :     16
Size of the tree :      31
```

Figura 5. Algoritmo J48: Árvore de decisão.

Matriz de Confusão - Árvore de Decisão					
		Classificação da DT			
		Sem vibração	Normal	Alerta	Crítico
Classe Real	Sem vibração	90	0	0	0
	Normal	0	88	2	0
	Alerta	0	1	89	0
	Crítico	0	0	0	90

Figura 6. Matriz de confusão: Árvore de decisão.

Tabela 1. Métricas de treinamento. Cálculos das *features*

Número de instâncias	360
Classificados corretamente	357
Classificados incorretamente	3
Acurácia	99.1667%

classificou a estrutura em estado crítico. A Figura 7 mostra a *dashboard* de monitoramento da vibração configurada na interface do software Unicleo®, que contém os gráficos dos sinais do acelerômetro no domínio do tempo e no domínio da frequência, as *features* calculadas pelo MLC e o resultado da classificação pela árvore de decisão. Neste caso, a classificação indicada é para o estado sem vibração. As Figuras 8, 9 e 10 mostram a classificação para os estados normal, em alerta e crítico da estrutura, respectivamente, comprovando o funcionamento do sistema e eficácia da árvore de decisão.

4. PROCESSO DE DETECÇÃO DE TRINCAS - SIMULAÇÃO

Pontos críticos em pontes, em que há carência de inspeção visual como complemento à análise de vibração, podem ser equipados com sistemas de aquisição e processamento de imagens para detecção de danos. A técnica consiste em obter imagens de pontos de interesse através de uma câmera e processá-las em uma CNN previamente treinada, substituindo de forma autônoma a inspeção visual.

4.1 Treinamento, testes e resultados

Para treinar a CNN e utilizá-la para classificar imagens de trincas, foi programado um algoritmo em linguagem Python® baseado nos códigos disponibilizados no repositório GitHub®, Rodrigues (2019). A prática consistiu no treinamento de uma CNN para classificar as imagens em duas classes distintas: "contém trincas" e "não contém trincas". O *dataset* utilizado no treinamento disponibiliza 40.000 imagens de superfícies de concretos, 20.000 das quais contêm trincas em diversos formatos, Ozgnel (2018). O projeto utilizou ainda as bibliotecas de *machine learning* TensorFlow® e Keras® para programação em Python®. O código do projeto foi dividido em 3 *scripts*. O primeiro converteu as imagens originais RGB em monocromáticas de dimensão 256x256 pixels e criou um arquivo único contendo todas as imagens e seus respectivos rótulos. Já o segundo efetuou o treinamento da CNN separando 20% dos dados para o conjunto de validação, onde utilizou a acurácia como métrica de performance e gerou, ao final, um arquivo contendo o modelo treinado, enquanto o terceiro utilizou este modelo para classificar quanto a presença ou não de trincas nas superfícies das imagens analisadas.

O processo de treinamento de redes neurais na modalidade *offline* é de alto custo computacional em termos de memória e processamento, devido ao tamanho em bytes do *dataset*, ao processo de escalar todas as imagens e torná-las monocromáticas, ao próprio treinamento, entre outros fatores, Krizhevsky et al. (2017). Assim, nesta etapa, optou-se pela utilização do Google Colab, que é um serviço de *notebook online*, da empresa Google, capaz de executar *scripts* em linguagem de programação Python® e também comandos para *Shell Linux*. A plataforma dispõe de Unidade de Processamento Gráfico (GPU) e Unidade de Processamento de Tensor (TPU) para processos que requerem maior velocidade de processamento. Na sessão estabelecida para o treinamento da CNN, o serviço disponibilizou 25.5 GB de memória RAM, CPU Intel® Xeon® 2.30 (GHz). Nesse trabalho, os treinamentos foram realizados com e sem aceleração por GPU. O tempo gasto para treinamento com GPU foi de 10 minutos e de 1 hora e 20 minutos com CPU. A Figura 11 mostra o diagrama em blocos de sequência de etapas do *script* de classificação de imagens da superfície com ou sem trincas. A imagem bruta é redimensionada e convertida de RGB para monocromática, em seguida é inserida na entrada da CNN que é composta por 3 camadas convolucionais configuradas igualmente com *kernels* 3x3 e função de ativação ReLU. As camadas de *max-polling* fazem a seleção das saídas de maior valor dentro de *kernels* 2x2, para diminuir a quantidade de cálculos da CNN. A camada *flatten* concatena as linhas da matriz da camada anterior, passando como entrada para a FCL um vetor de features de 1 dimensão. A camada FCL é configurada com a função de ativação sigmoide para gerar os resultados da classificação.

Uma das métricas utilizadas para desempenho da CNN é a acurácia, isto é, a taxa de predições corretas sobre o conjunto de teste do *dataset*. Para que a CNN possa alcançar boa acurácia, são necessárias múltiplas repetições

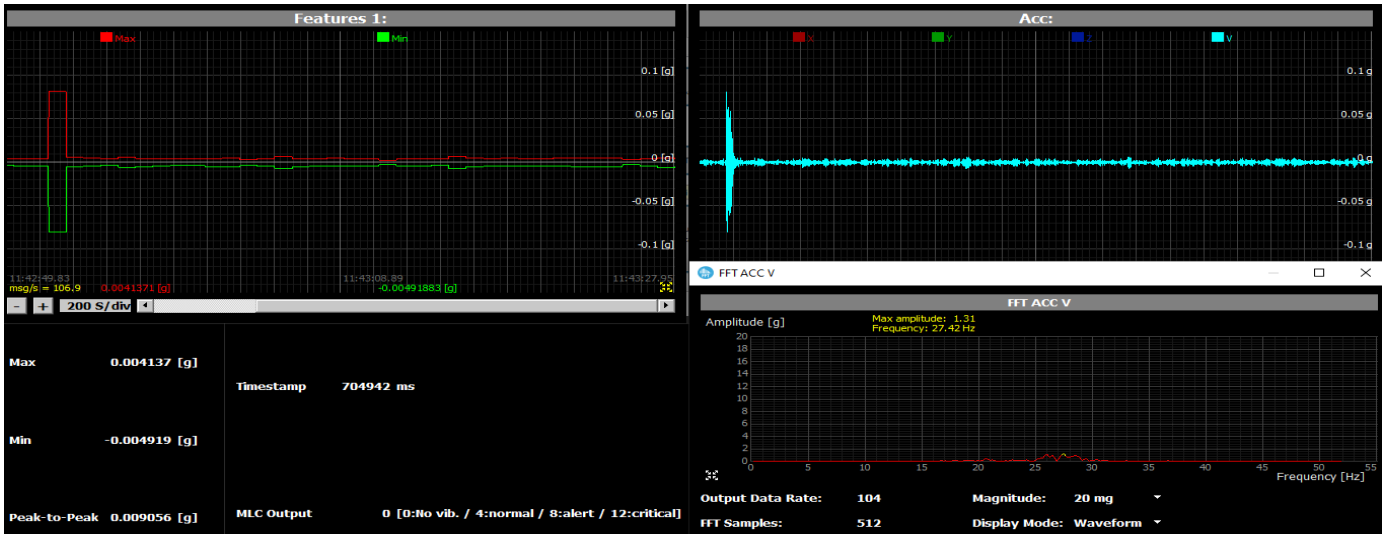

Figura 7. *Dashboard* de monitoramento de vibração, classificando o estado da estrutura como sem vibração.

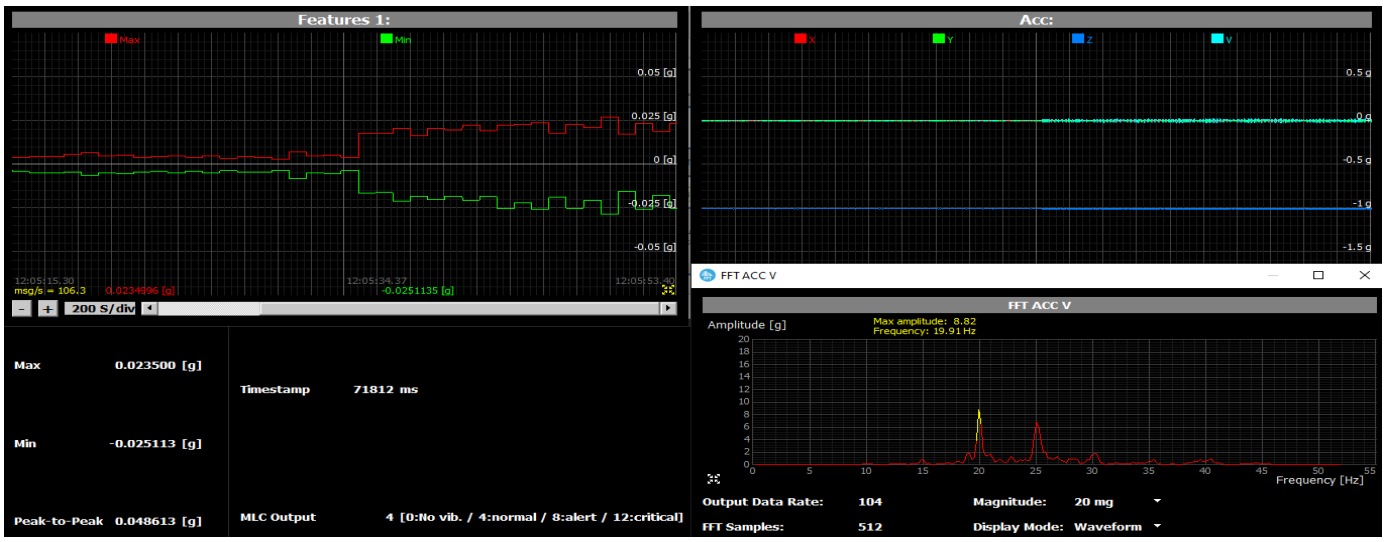

Figura 8. *Dashboard* de monitoramento de vibração, classificando o estado da estrutura como normal.

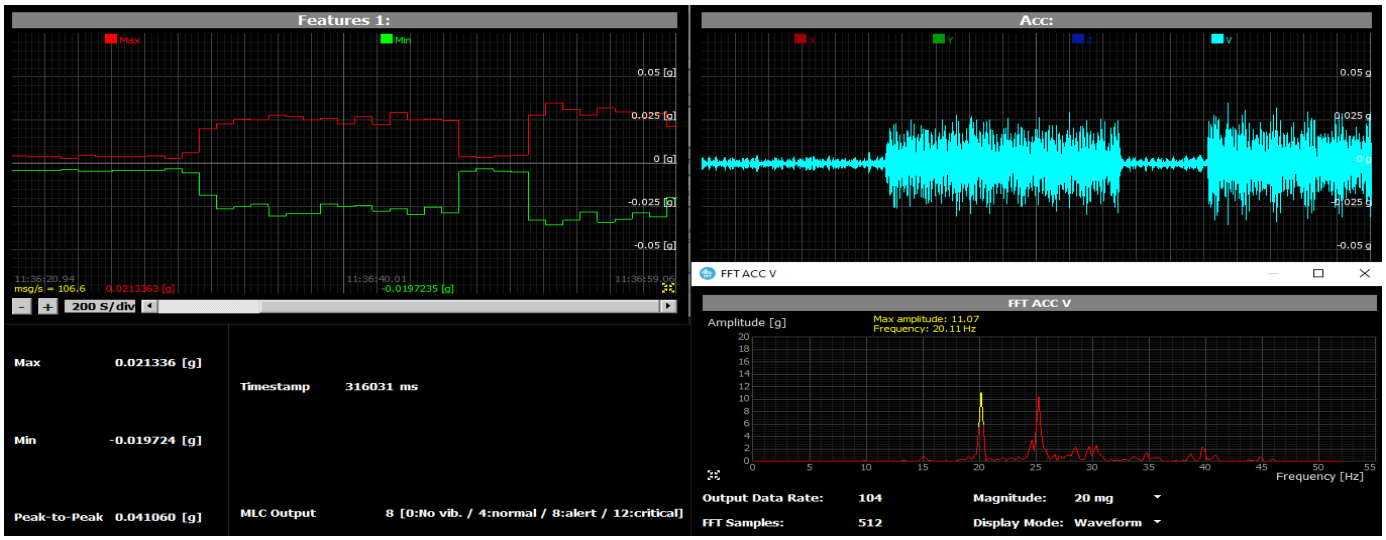

Figura 9. *Dashboard* de monitoramento de vibração, classificando o estado da estrutura como em alerta.

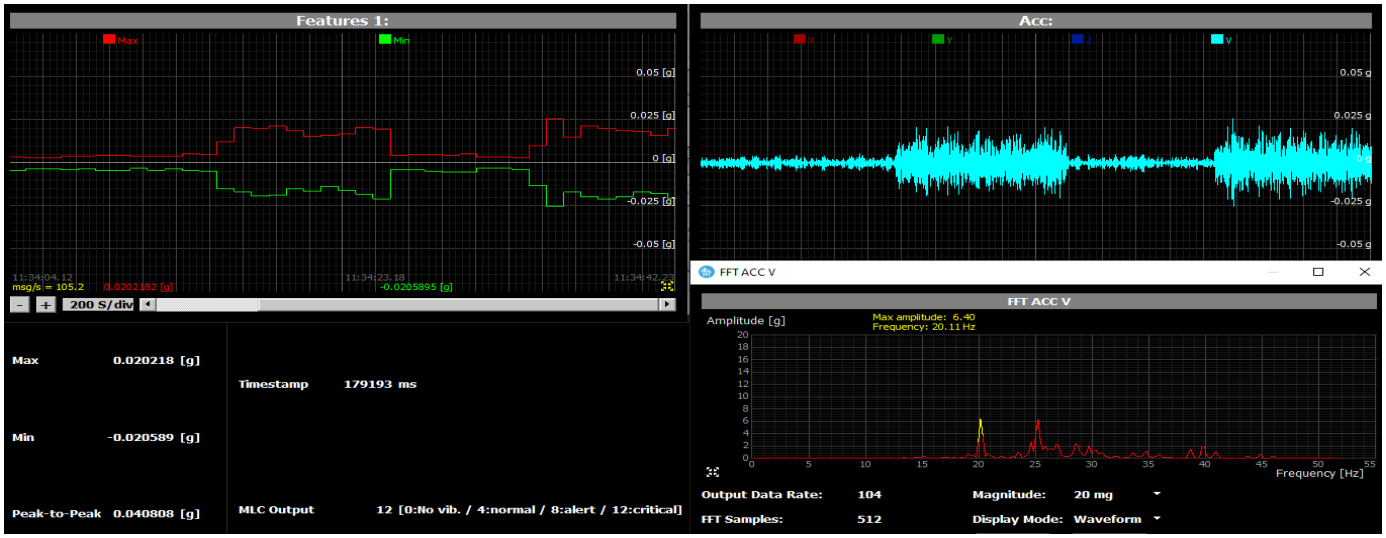

Figura 10. *Dashboard* de monitoramento de vibração, classificando o estado da estrutura como crítico.

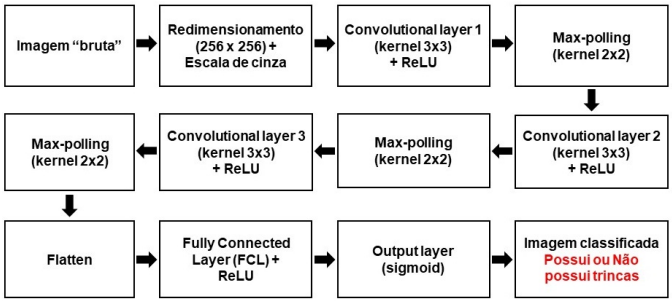

Figura 11. Etapas do *script*: Rede Neural Convolucional.

do treinamento sobre todo o conjunto de treino do *dataset*, denominadas épocas. Conforme apresentado na Figura 12, em um treinamento com 5 épocas (0 à 4), foi possível atingir uma acurácia de 0.9930 no conjunto de treinamento e 0.9926 no conjunto de validação.

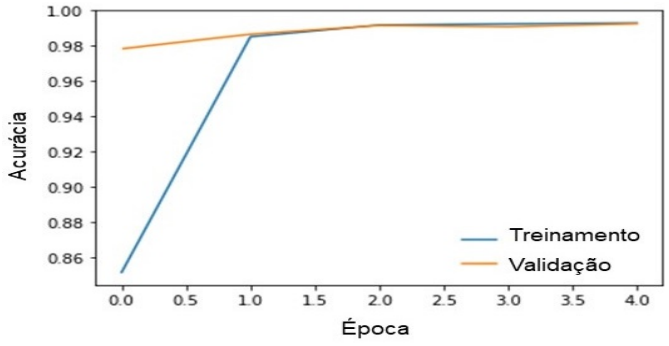

Figura 12. Época x Acurácia: Evolução da acurácia da CNN.

Além do conjunto de testes extraído do *dataset*, foram feitos testes com imagens capturadas por uma câmera de um *smartphone*. Para esse teste, as imagens capturadas foram diferentes dos padrões encontrados no *dataset* utilizado, visando avaliar a capacidade de generalização atingida pela CNN. O novo conjunto de teste é composto por imagens de muros residenciais de concreto em pontos com e sem trincas, totalizando 29 imagens das quais 15 com trincas. A Figura 13 ilustra e compara imagens de ambos os conjuntos de teste. Em (a) são mostradas as imagens com trincas do *dataset*, em (b) as imagens com trincas capturadas por um *smartphone*, em (c) as imagens sem trinca do *dataset* e em (d) as imagens sem trincas capturadas por um *smartphone*. É possível notar as diferenças de texturas, iluminação e tamanho da área da superfície das imagens enquadradas.

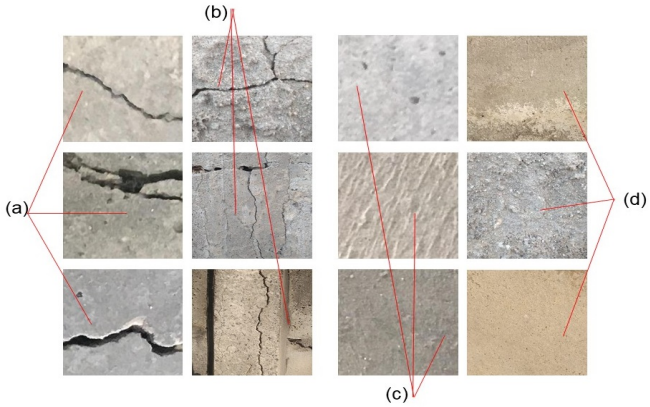

Figura 13. Images: *Dataset* e *smartphone*.

A métrica de desempenho utilizada no novo conjunto de teste foi a matriz de confusão da Figura 14, que além de permitir a análise do desempenho da CNN como um todo, também possibilita analisá-la em cada classe separadamente, quantificando predições corretas, incorretas e quais foram as classes confundidas nas predições incorretas. A diagonal principal da matriz de confusão representa as predições corretas, portanto na condição ótima os valores da diagonal principal são maximizados e os demais valores minimizados. Um pequeno *dataset* contendo 29 imagens foi usados para teste da CNN, das quais 15 continham trincas e todas foram classificadas corretamente, 12 não continham trincas e foram também classificadas também corretamente e 2 não continham trincas e foram classificadas incorretamente. Nas predições incorretas foi observado que as imagens continham uma textura com aspecto áspero e granulado, sendo esta característica não constante no conjunto de treinamento. A Figura 15 apresenta as 2 predições incorretas.

		Matriz de Confusão - Classificação da CNN	
		Com trinca	Sem trinca
Classe Real	Com trinca	15	0
	Sem trinca	2	12

Figura 14. Matriz de confusão: Rede Neural Convolucional.

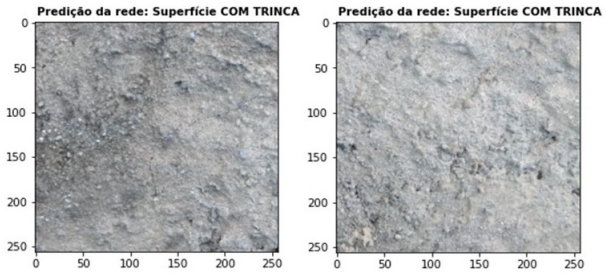

Figura 15. Imagens do *smartphone*: 2 predições incorretas.

5. CONCLUSÕES

O sistema desenvolvido e testado para o monitoramento de vibração em estruturas civis, mostrou-se promissor para a visualização dos dados de vibração no domínio do tempo e da frequência, bem como, para a classificação de estados pré-determinados, por meio do treinamento supervisionado de uma árvore de decisão.

O modelo de CNN utilizado na detecção de trincas por imagem neste trabalho apresentou resultados satisfatórios, indicando possibilidades promissoras para a implementação em um protótipo de sistema embarcado dotado com uma câmera e testes em pontes reais.

REFERÊNCIAS

Aaron, C., Ian, G. e Yoshua, B. (2016). Chapter 9: convolutional networks. In: *Deep learning*. Cambridge: MIT press, 365-366.

Avci, O., Abdeljabemr O., Kiranyaz, K. e Hussein, M. (2018). Wireless and real time structural damage detection: A novel decentralized method for wireless sensor networks. *Journal of Sound and Vibration*:158-172. DOI: https://doi.org/10.1016/j.jsv.2018.03.008.

Abdulkarem, M., Samsudin, k. e Zaman, F. (2019). Wireless sensor network for structural health monitoring: A contemporary review of technologies, challenges, and future direction. *Structural Health Monitoring*:1–43. DOI:https://doi.org/10.1177/1475921719854528.

Aggarwal, C. (2018). Neural Networks and Deep Learning. Available at: http://dx.doi.org/10.1007/978-3-319-94463-0.

Bachmann, H., Ammann, W. e Deischl, F. (1994). *Vibration Problems in Structures Practical Guidelines*. Birkhäuser Verlag.

Beheshti, S., Ahmadian, V. e Malda,r M. (2020). Damage detection of structures using signal processing and artificial neural networks. *Advances in Structural Engineering*:892–896. DOI: https://doi.org/10.1177/1369433219886079.

Fernandez, C. , Lozano, R. e Villatoro, R. (2017). Efficient pavement crack detection and classification. *EURASIP Journal on Image and Video Processing*:20-39. DOI:10.1186/s13640-017-0187.

Gunasegarana, V. e Muralidharana, V. (2019). Fault Diagnosis of Spur Gear System through Decision Tree Algorithm Using Vibration Signal. *ScienceDirect*:3233-3239. DOI:https://doi.org/10.1016/j.matpr.2020.03.283.

Jang, K., Kim, N. e An, Y-K. (2019). Deep learning based autonomous concrete crack evaluation through hybrid image scanning. *Structural Health Monitoring*:1722-1737. DOI:10.1177/1475921718821719.

Jesus, W., Lima, E. e Siva, J. (2019).Vibration and position monitoring in civil structures using Wi-Fi systems. *Set international journal of broadcast engineering*:96. DOI:http://dx.doi.org/10.18580/setijbe.2019.12.

Krizhevsky, A., Sutskever, Ilya e Hinton, G. (2017). ImageNet Classification with Deep Convolutional Neural Networks. *Commun. ACM*:84–90. DOI: https://doi.org/10.1145/3065386.

Li, Q., Cai, W. e Wang, X. (2019). Medical image classification with convolutional neural network. *13th International Conference on Control Automation Robotics Vision*:51–66. DOI: 10.1109/ICARCV.2014.7064414.

Liu, J., Yang, X. e Li, L. (2019). Vibronet: Recurrent neural networks with multi target learning for image-based vibration frequency measurement. *Journal of Sound and Vibration*:51–66. DOI:https://doi.org/10.1016/j.jsv.2019.05.027.

LSM6DSOX. Available at: https://www.st.com/en/mems-and-sensors/lsm6dsox.html#overview (accessed 16 May 2020).

Ozgnel, F. Concrete Crack Images for Classification. 2018. Available at: http://dx.doi.org/10.17632/5y9wdsg2zt.1#file-c0d86f9f-852e-4d00-bf45-9a0e24e3b932 (accessed 22 May 2020).

Quinlan, J. (2014) *C4.5: Programs for Machine Learning*. London:Cassell.

Rodrigues, B. Concrete Crack Classification with Tensorflow Keras API. 2019. Available at: https://github.com/vbrodrigues/Concrete-Crack-Classification-Model(accessed 22 May 2020).

Sharma, A., Sugumaran, V. e Devasenapati, S. (2014). Misfire detection in an IC engine using vibration signal and decision tree algorithms. *Measurement*:370-380.DOI:https://doi.org/10.1016/j.measurement.2014.01.018.

Serway, R. e Jewett, J. (2018). *Physics for Scientists and Engineers*. 10th ed.: Cengage Learning.

Shimoi, N., Saijo, M. e Cuadra, C. (2015). Comparison of Natural Frequency Vibration Analysis for a Bridge Using Accelerometers and a Piezoelectric Cable Vibration Sensor.*International Journal of Instrumentation Science*:1-9. DOI:10.5923/j.instrument.20150401.01.

Subramanian, N. (2013). *Design of structured concrete structures*.Oxford.

Wei, X., Jin, T. e Chen, P. (2019). Structural crack detection using deep learning–based fully convolutional networks.*Advances in Structural Engineering*:3412–3419. DOI: https://doi.org/10.1177/1369433219836292.

Weimer, D., Scholz, B. e Shpitalni, S. (2015). Design of deep convolutional neural network architectures for automated feature extraction in industrial inspection. *CIRP Annals - Manufacturing Technology*: 417-418. DOI: https://doi.org/10.1016/j.cirp.2016.04.072.