UNIVERSIDADE FEDERAL DE JUIZ DE FORA
FACULDADE DE ENGENHARIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA CIVIL

Luiz Tadeu Dias Júnior

Uso de autocodificadores variacionais para a detecção de danos estruturais

Juiz de Fora
2024

Luiz Tadeu Dias Júnior

Uso de autocodificadores variacionais para a detecção de danos estruturais

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia Civil da Faculdade de Engenharia da Universidade Federal de Juiz de Fora como requisito parcial à obtenção do título de Mestre em Engenharia Civil. Área de concentração: Estruturas e Materiais. Linha de pesquisa: Mecânica das estruturas.

Orientador: Prof. Dr. Alexandre Abrahão Cury

Coorientadores: Prof. Dr Flávio de Souza Barbosa e Dr.ª Rafaelle Piazzaroli Finotti Amaral.

Juiz de Fora

2024

Ficha catalográfica elaborada através do Modelo Latex do CDC da UFJF
com os dados fornecidos pelo(a) autor(a)

Dias Júnior, Luiz Tadeu.
Uso de autocodificadores variacionais para a detecção de danos estruturais / Luiz Tadeu Dias Júnior. – 2024.
98 f. : il.

Orientador: Prof. Dr. Alexandre Abrahão Cury
Coorientadores: Prof. Dr. Flávio de Souza Barbosa e Dr.ª Rafaelle Piazzaroli Finotti Amaral
Dissertação (mestrado acadêmico) – Universidade Federal de Juiz de Fora, Faculdade de Engenharia. Programa de Pós-Graduação em Engenharia Civil, 2024.

1. Dinâmica das estruturas. 2. Monitoramento de integridade estrutural. 3. Detecção de danos. 4. Aprendizado de Máquina. 5. Autocodificadores Variacionais. I. Cury, Alexandre Abrahão, orient. II. Barbosa, Flávio de Souza, coorient. III. Amaral, Rafaelle Piazzaroli Finotti, coorient. IV. Título

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA CIVIL

LUIZ TADEU DIAS JUNIOR

Título: *"Uso de autocodificadores variacionais para a detecção de danos estruturais."*

Dissertação apresentada ao Programa de Pós-graduação em Engenharia Civil da Universidade Federal de Juiz de Fora como requisito parcial à obtenção do título de Mestre em Engenharia Civil. Área de concentração: Estruturas e Materiais.

Aprovada em 26 de fevereiro de 2024.

BANCA EXAMINADORA

Prof. Dr. Alexandre Abrahão Cury - Orientador e Presidente da Banca
Universidade Federal de Juiz de Fora

Prof. Dr. Flávio de Souza Barbosa - Coorientador
Universidade Federal de Juiz de Fora

Prof. Dr. Yuri José Oliveira Moraes - Membro Titular Interno
Universidade Federal de Juiz de Fora

Prof. Dr. Rafael Holdorf Lopez - Membro Titular Externo

Juiz de Fora, 07/02/2024.

Documento assinado eletronicamente por **Alexandre Abrahao Cury**, **Professor(a)**, em 26/02/2024, às 10:43, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Yuri José Oliveira Moraes**, **Professor(a)**, em 26/02/2024, às 15:11, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Rafael Holdorf Lopez**, **Usuário Externo**, em 26/02/2024, às 19:45, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Flavio de Souza Barbosa**, **Professor(a)**, em 27/02/2024, às 08:21, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade deste documento pode ser conferida no Portal do SEI-Ufjf (www2.ufjf.br/SEI) através do ícone Conferência de Documentos, informando o código verificador **1699373** e o código CRC **4324611E**.

AGRADECIMENTOS

Dedico meus agradecimentos:

- Em primeiro lugar a Deus, pois sem ele nada é possível, me guiou e protegeu ao longo dessa jornada;
- Aos meus orientadores e coorientadores, Alexandre Cury, Flávio Barbosa e Rafaelle Finotti, por todo o incentivo, ensinamento, atenção e confiança em mim depositados;
- A todos os professores e funcionários do Programa de Pós-graduação em Engenharia Civil da UFJF;
- Aos amigos e companheiros de trabalho do mestrado, em especial ao Matheus Dalcin, Ricardo Vidal e Júlia Kunsch, que me auxiliaram durante esses dois anos;
- Aos meus pais e irmãs, Luiz Tadeu, Lucilene, Lidiane e Deise, por todo apoio;
- À minha namorada, Ariádine, por todo incetivo e cuidado;
- À Capes, pelo suporte financeiro que foi essencial para que eu prosseguisse no programa;
- À UFJF, por toda a infraestrutura disponibilizada ao longo do meu mestrado, ratificando seu compromisso em entregar um ensino público, gratuito e de qualidade.

RESUMO

O estado de integridade de uma estrutura, bem como o seu monitoramento, têm sido constantemente pauta de estudo, dados os avanços tecnológicos mundiais nesta área. Assim, surge o conceito de Monitoramento da Integridade Estrutural (SHM, do inglês *Structural Health Monitoring*), como sendo um conjunto de técnicas e equipamentos utilizados para investigar o estado de integridade de estruturas. Nesse contexto, com a rápida evolução tecnológica computacional e de informação, algoritmos de inteligência artificial focados no aprendizado profundo têm ganhado cada vez mais espaço no que tange à análise de comportamentos estruturais baseados em suas respostas dinâmicas. Dentro desse escopo, o presente trabalho se dedica à avaliação de um algoritmo de aprendizado profundo denominado Autocodificador Variacional (VAE) quando utilizado como extrator de características de dados dinâmicos. Desse modo, avalia-se a capacidade dos modelos VAE de fornecer conjuntos de variáveis que viabilizem a detecção de um comportamento considerado anormal para uma dada estrutura. A metodologia desenvolvida consiste na reconstrução dos sinais dinâmicos utilizando modelos VAE. A partir dos erros de reconstrução, duas métricas são utilizadas para discriminar diferentes cenários de danos nas estruturas analisadas: o MSE (Mean Square Error) e uma combinação do MSE com a Distância de Mahalanobis. Duas aplicações experimentais reais são utilizadas para validar a abordagem proposta. Inicialmente, uma estrutura de pequena escala: um pórtico plano ensaiado no Laboratório de Imagens e Sinais da UFJF; em seguida, uma estrutura em escala real: a clássica ponte Z24, na Suíça. Os resultados obtidos para o pórtico foram bastante satisfatórios, sendo a metodologia capaz de distinguir os diversos comportamentos estruturais com uma acurácia variando perto dos 99%. Entretanto, para o caso da ponte Z24, houve casos em que a classificação não se mostrou totalmente adequada, o que indica a dificuldade de se trabalhar diretamente com dados brutos de vibração para a detecção de comportamentos anormais em estruturas.

Palavras-chave: Dinâmica das estruturas. Monitoramento de integridade estrutural. Detecção de danos. Aprendizado de Máquina. Autocodificadores Variacionais.

ABSTRACT

The state of integrity of a structure, as well as its monitoring, have constantly been the subject of study, given global technological advances in this area. Thus, the concept of Structural Health Monitoring (SHM) emerges, as a set of techniques and equipment used to investigate the state of integrity of structures. In this context, with the rapid evolution of computational and information technology, artificial intelligence algorithms focused on deep learning have been gaining more and more space when it comes to analyzing structural behaviors based on their dynamic responses. Within this scope, the present work is dedicated to the evaluation of a deep learning algorithm called Variational Autoencoder (VAE) when used as a feature extractor from dynamic data. In this way, the ability of VAE models to provide sets of variables that enable the detection of behavior considered abnormal for a given structure is evaluated. The methodology now developed consists of reconstructing dynamic signals using VAE models. From the reconstruction errors, two metrics are used to discriminate different damage scenarios in the analyzed structures: the MSE (Mean Square Error) and a combination of the MSE with the Mahalanobis Distance. Two real experimental applications are used to validate the proposed approach. Initially, a small-scale structure: a flat portico tested at the UFJF Images and Signals Laboratory; then a full-scale structure: the classic Z24 bridge in Switzerland. The results obtained for the frame were quite satisfactory, with the methodology capable of distinguishing the different structural behaviors with an accuracy varying close to 99%. However, in the case of the Z24 bridge, there were cases in which the classification was not completely adequate, which indicates the difficulty of working directly with raw vibration data to detect abnormal behavior in structures.

Keywords: Dynamics of structures. Structural health monitoring. Damage detection. Machine learning. Variational Autoencoder.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AE	*Autocodificador*
Bi-LSTM	*Bidirectional Long Short-Term Memory*
DL	*Aprendizado profundo. Do inglês, Deep Learning*
LIS	*Laboratório de Imagens e Sinais*
LSTM	*Long Short-Term Memory*
MLP	*Multiplayer Perceptron*
MSE	*Mean Squared Error*
FC	*Totalmente conectada. Do inglês, fully conected*
RNAs	*Redes Neurais Artificiais*
RNN	*Redes Neurais Recorrentes*
SAE	*Autocodificador Esparso*
SHM	*Structural Health Monitoring*
VAE	*Autocodificador Variacional*

SUMÁRIO

1 INTRODUÇÃO

1.1 CONTEXTO E MOTIVAÇÃO

A segurança, alinhada ao adequado funcionamento de uma estrutura, são algumas das principais preocupações dos engenheiros ao longo da vida útil das construções. Nesse sentido, erros na concepção da estrutura, falhas durante a etapa da execução, efeitos ambientais, desgastes naturais ou até mesmo eventuais danos pontuais sofridos por ela, são algumas das causas que podem danificá-la. Desse modo, tais alterações podem gerar comportamentos dinâmicos atípicos que podem estar atrelados ao surgimento de danos estruturais [1] [2].

Nesse contexto, tornou-se fundamental estabelecer critérios e técnicas que permitissem realizar o monitoramento da integridade estrutural de uma edificação. Essa metologia é mundialmente conhecida pela sua sigla em inglês SHM (*Structural Health Monitoring*), a qual envolve a observação e a análise de um sistema estrutural ao longo do tempo. O SHM é um campo de estudo que se concentra na avaliação contínua das estruturas, de modo a assegurar a sua integridade, o seu desempenho e a sua segurança. Assim, tal metodologia é baseada no uso de várias tecnologias e equipamentos associados a técnicas de análise de dados e modelos preditivos para monitorar e avaliar a integridade de estruturas.

Compreender a importância do estudo do SHM é crucial, haja vista que desempenha um papel fundamental no aumento da vida útil, confiabilidade e resiliência da edificação. Viadutos e pontes, por exemplo, são essenciais para uma sociedade, pois servem como meios de conexão e de transporte. Entretanto, tais estruturas estão sujeitas a vários mecanismos de degradação. Nesse contexto, com o uso do SHM, engenheiros e pesquisadores se tornaram aptos a desenvolver técnicas capazes de detectar danos ou alterações estruturais. Além disso, o SHM permite realizar manutenções preventivas e/ou reparos, assegurando não só a segurança do usuário, como também reduzindo os custos [1].

Um caso próximo que ilustra bem a importância (da falta) de um sistema de SHM na prática é o do rompimento de uma barragem de rejeitos de minério de ferro no distrito de Bento Rodrigues do município de Mariana, no estado de Minas

Gerais (Figura 1). A lama espalhada trouxe consequência graves à fauna e à flora do local, além de ter sido responsável pela morte de dezenas pessoas, resultando no maior desastre socioambiental brasileiro.

Figura 1 – Antes e depois do rompimento da barragem no distrito de Bento Rodrigues

Fonte: Lopes e Dias [3]

Portanto, o estudo de técnicas de SHM permite não só evitar catástrofes, como também otimizar estratégias de manutenção. As tradicionais práticas de manutenção geralmente empregadas envolvem inspeções, programas ou reparos à medida que se fazem necessários com base em um certo cronograma ou após a ocorrência de uma falha. Esse tipo de abordagem pode resultar em prejuízos, tendo em vista que geram tempo de inatividade da estrutura ou reparos desnecessários. Desse modo, o SHM oferece uma abordagem bem mais sistêmica e orientada com base em dados, já que pode monitorar continuamente uma estrutura. Logo, os tomadores de decisão podem fazer escolhas mais bem informadas sobre intervenções para manutenção, otimizando recursos e minimizando a interrupção da atividade da edificação.

Nesse sentido, é evidente a importância da validação e do desenvolvimento de sistemas de monitoramento que auxiliem na detecção de danos de uma estrutura.

No mundo, programas de monitoramento contínuo têm sido cada vez mais utilizados em estruturas de grande porte, como é o caso da Ponte Rio-Niterói no Brasil [4], a ponte Tsing Ma, na China [5], o viaduto de Millau, na França [6], dentre outros. Segundo Riera [7], mudanças naturais de material ou até mesmo pequenos danos são previstos na literatura, mas ainda assim devem ser analisados com devido cuidado em grandes edificações e, de preferência, com monitoramento contínuo. Isso nos mostra a importância do SHM sobretudo em edificações de grande porte, ainda que se observe as normas brasileiras [8] [9] [10] e/ou internacionais [11] [12].

Nos últimos anos, com a evolução da tecnologia computacional e de informação, grandes avanços foram alcançados nos sistemas de aquisição de dados, o que, por consequência, aprimoraram as ferramentas e as técnicas de SHM para lidar com o grande volume de dados gerados [13]. Assim, o que antes era feito com um número reduzido de variáveis por meios probabilísticos, técnicas estatísticas e análise modal, passou a ser realizado automaticamente com o uso de métodos de aprendizado de máquina (*Machine Learning*) [14] [15]. Dentre eles, destaca-se as Redes Neurais Artificias (RNAs).

As RNAs são modelos computacionais inspirados na estrutura e no funcionamento de redes neurais biológicas do cérebro humano [16]. Assim, são capazes de aprender, separar e classificar dados. Para tal, as redes neurais são compostas de nós interconectados referidos como neurônios. Apesar de existirem diferentes tipos de RNAs, o presente trabalho foca nos Autocodificadores (AE, do inglês *Autoencoder*). Essa atenção especial é justificável dado os resultados apresentados em pesquisas, a serem mostradas mais adiante, utilizando esse tipo de arquitetura.

O Autocodificador é uma técnica de aprendizado profundo (DL, do inglês *deep learning*) comumente utilizada para tarefas de detecção de anomalias, extração de recursos e construção de dados [17]. Assim, tal RNA é um modelo de aprendizado não-supervisionado (conjunto de dados que não precisam de uma pré-rotulação [18] [19]). Em resumo, o AE é um modelo preditivo que trabalha com a codificação dos dados de entrada e, *a posteriori*, a reconstrução dos dados originais a partir dessa codificação. Assim, a arquitetura básica de um Autocodificador consiste em um codificador e um decodificador. O primeiro recebe os dados de entrada e os mapeia para uma representação de dimensão inferior (denominada espaço latente)

e o segundo recebe essa representação codificada e reconstrói os dados originais.

O presente trabalho foca em avaliar o uso do Autocodificador Variacional (VAE) aplicado ao monitoramento de integridade estrutural. O VAE é um modelo que combina elementos de um Autocodificador e um modelo probabilístico, permitindo aprender e gerar novas amostras de dados [20].

1.2 OBJETIVOS

O principal objetivo desta dissertação consiste em avaliar a aplicação do VAE em problemas de SHM de estrutura civis. Assim, serão analisadas as variáveis que definem o comportamento do VAE, além da capacidade de esta técnica reproduzir sinais dinâmicos em uma abordagem não supervisionada. Os objetivos específicos são:

- Avaliar a sensibilidade dos hiperparâmetros associados ao VAE na reconstrução das respostas dinâmicas;
- Estimar o custo computacional, em termos de tempo de processamento, para se avaliar o uso do VAE em casos de monitoramento dinâmico em tempo real;
- Avaliar a capacidade do VAE de identificar diferentes níveis de danos em estruturas;
- Avaliar o impacto do uso de diferentes tipos de dados de entrada (sinais no domínio do tempo e da frequência) nos resultados obtidos.

1.3 ORGANIZAÇÃO DO TRABALHO

Este trabalho está estruturado nos seguintes capítulos:

- Capítulo 1: Apresenta a relevância do tema, a motivação para se realizar este estudo e a descrição dos objetivos que se pretende alcançar;
- Capítulo 2: Mostra uma revisão histórica e bibliográfica sobre diversos estudos disponíveis na literatura e como esses trabalhos contribuíram para o tema em questão;

- Capítulo 3: Descreve a metodologia empregada neste estudo, bem como as definições para a compreensão e o desenvolvimento da pesquisa;
- Capítulo 4: Apresenta as aplicações que foram feitas com o uso do método proposto bem como seus resultados;
- Capítulo 5: Expõe as conclusões e as considerações finais, além de sugestões para pesquisas futuras.

2 REVISÃO DA LITERATURA

2.1 ESTADO DA ARTE

Nos últimos anos, pesquisadores têm sido encorajados pelas novas possibilidades de estudo do SHM utilizando-se aprendizado de máquina. Assim, o presente capítulo busca abordar de forma sintética os principais estudos desenvolvidos nesta área nos últimos anos.

2.1.1 Métodos baseados em parâmetros modais e indicadores evoluídos

Uma das principais metodologias empregadas no início do estudo da avaliação de alterações estruturais / detecção de danos foram aquelas desenvolvidas com base na variação dos parâmetros modais, isto é, das frequências naturais, das taxas de amortecimento e dos modos de vibração, haja vista a relação direta entre a resposta dinâmica de um material com as suas propriedades mecânicas. Resumidamente, dado que o dano altera as propriedades físicas da estrutura, como a sua massa e a sua rigidez, logo poderá alterar também os seus parâmetros modais. Portanto, se há alguma modificação nesses parâmetros, pode ser que a estrutura analisada esteja com algum tipo de dano [14] [15].

Nos primórdios desses estudos, a detecção de danos era baseada na avaliação direta das propriedades modais. As frequências naturais, por serem indicadores sensíveis ao estado de integridade de uma estrutura tendem a apresentar valores ligeiramente menores em situações em que a estrutura encontra-se danificada. Em outras palavras, como o dano conduz a uma perda da rigidez localizada, é de se esperar que o valor da frequência natural diminua. Cawley e Adams [21] foram os pioneiros na avaliação da integridade estrutural utilizando a variação das frequências naturais como indicador de dano. A partir desse trabalho, novos estudos foram realizados seguindo a mesma linha, como Fox [22] que apresentou resultados numéricos e experimentais de uma viga fissurada e mostrou a sensibilidade das frequências naturais como indicadores de danos. Mais recentemente, os autores Sha *et al.* [23] mostraram ser possível visualizar em uma viga fissurada não só o dano, como também o local onde ele se encontra, através de relações entre as curvas de

mudança das frequências.

Outro parâmetro modal fartamente encontrado na literatura utilizado como indicador para detecção de danos são as variações das amplitudes dos modos de vibração. Um dos métodos mais antigos e clássicos encontrados é o MAC (*Modal Assurance Criterion*), proposto por Allemang [24]. Esse índice calcula a correlação entre modos danificados e não-danificados, variando de 0 a 1, onde 0 representa a inexistência de correlação entre os modos e 1 uma correlação perfeita. Seguindo a mesma linha de raciocínio, os autores Lieven e Ewins [25] apresentaram o COMAC (*Coordinate Modal Assurance Criterion*), que mede a correlação dos modos para cada grau de liberdade do sistema estrutural. Nesse caso, se o índice é igual a 1, significa uma correlação perfeita entre as amplitudes de uma dada coordenada.

Nesse contexto, diversos estudos sobre a aplicação direta dos parâmetros modais destacaram a limitada sensibilidade das frequências naturais, revelando ao mesmo tempo um potencial promissor na utilização dos modos próprios para detecção e/ou localização de danos. Assim, diversas abordagens foram propostas para aumentar a capacidade dos indicadores baseados nos modos próprios para detecção de danos [26] [27] [28].

Em 1991, Pandey, Biswas e Samman [29] apresentaram um indicador baseado na variação da curvatura dos modos naturais de estruturas saudáveis e danificadas, associando a mudança na rigidez de flexão à mudança na curvatura. Dois anos depois, Kim e Stubbs [30], apresentaram um método baseado na variação de energia de deformação. Nesse caso, analisou-se esse parâmetro antes e depois da ocorrência de um dano e os resultados mostraram que era possível detectar e localizar a região afetada da estrutura. Além disso, para que fosse possível analisar, identificar e localizar danos estruturais através da matriz de flexibilidade, os autores Pandey e Biswas [31] mostraram que é necessário associar a presença do dano à redução de rigidez e ao aumento da flexibilidade. Nesse sentido, um estudo no qual diversos desses métodos baseados em indicadores evoluídos foram analisados. Alvandi e Cremona [32], por exemplo, mostraram que o método baseado na variação da energia de deformação se apresentou como sendo o mais eficaz para localizar o dano, mas não para quantificá-lo. Já os métodos baseados na mudança de curvatura dos modos e na análise de flexibilidade permitiram quantificar o dano. Em 2011,

os autores Cury, Borges e Barbosa [33] apresentaram uma estratégia que uniu a variação da energia de deformação e as frequências naturais para localizar e quantificar o dano, obtendo bons resultados.

Ainda que as análises das mudanças de parâmetros modais sejam abordagens capazes de detectar e quantificar o dano em uma estrutura, a obtenção dessas propriedades pode ser afetada por diversos fatores, tanto ambientais (temperatura, vento e radiação solar), quanto operacionais (carregamentos de utilização), prejudicando o uso dessas metodologias em casos práticos de SHM [14] [15].

2.1.2 Métodos baseados em inteligência artificial

Os procedimentos apresentados nas seções anteriores fundamentam-se principalmente em uma análise temporal para extrair informações por meio de características modais específicas ou indicadores evoluídos derivados delas. Embora esses indicadores tenham revelado as suas eficácias, também evidenciaram alguns desafios, como a baixa sensibilidade para certos níveis de dano, a dependência de um estado de referência e questões relacionadas à confiabilidade, especialmente em termos da probabilidade de detecção de alarmes falsos. Um ponto crucial a ser levado em consideração é que, para esses métodos, presume-se a hipótese de que o comportamento mecânico é linear após danos, o que nem sempre é verídico. Além disso, o processo de identificação modal funciona como um filtro, podendo resultar na perda de informações essenciais sobre o estado da estrutura. Simultaneamente, há uma escassez de pesquisas direcionadas à análise direta da resposta dinâmica de uma estrutura mediante as suas características sob excitações externas. De fato, os sinais obtidos a partir de medições dinâmicas são pouco empregados devido à falta de ferramentas adequadas para manipulação e à dificuldade associada ao uso direto desses sinais [27].

Entretanto, com o crescente avanço tecnológico, pesquisas nesse tema aumentaram vertiginosamente nos últimos anos, já que possibilitou tratar maiores conjuntos de dados, reduzindo-se a perda de informação [27]. Assim, alternativamente aos métodos baseados exclusivamente em parâmetros modais, surgem técnicas de monitoramento baseadas no reconhecimento de formas (*pattern recognition*), também conhecido como reconhecimento de padrões. O reconhecimento

de padrões, aplicado à engenharia civil, consiste na detecção de comportamentos estruturais considerados anômalos. Desta forma, os sinais brutos (no domínio do tempo ou no da frequência) são processados através de técnicas estatísticas, sendo comparados e classificados por algoritmos de inteligência computacional para identificar possíveis anomalias no estado estrutural. Nesses casos, as respostas dinâmicas obtidas pelo SHM podem ser processadas através do aprendizado de máquina (ou do inglês, *Machine Learning*).

Chang *et al.* [34] apresentaram um trabalho que detecta danos baseados em Redes Neurais Iterativas e no conceito de matriz ortogonal. A finalidade desse estudo gerou combinações representativas das alterações de parâmetros. Desse modo, a partir de testes numéricos conseguiram comprovar a eficiência do método na identificação de danos.

No trabalho de Iwasaki *et al.* [35] foram apresentados métodos de agrupamentos (*clustering*) para detecção de danos em vigas a partir de diferentes respostas no domínio da frequência. Os resultados das classificações mostram que o algoritmo consegue identificar os diferentes níveis de danos impostos à estrutura. Nas pesquisas de Alves *et al.* [36] e Cury [26] foi possível avaliar o desempenho de técnicas estatísticas de Análise dos Dados Simbólicos, apresentando resultados úteis em relação à compreensão e à representação dos sinais.

Em trabalhos mais recentes, Ma *et al.* [37] publicaram uma metodologia que utiliza aprendizado profundo para detectar, com eficácia, danos estruturais. Tal estudo utiliza como base um Autocodificador Variacional associado à uma aprendizagem não supervisionada. De modo a validar o estudo, a abordagem foi aplicada a uma ponte em funcionamento, da qual obtiveram resultados que comprovam que o método proposto pode identificar com precisão os danos estruturais. Ademais, Pollastro *et al.* [38] apresentaram métodos semi-supervisionados com uma abordagem baseada em dados para detecção de anomalias utilizando Redes Neurais Artificiais e Autocodificador Variacional. Posteriormente, os autores utilizaram Máquina de Vetores de Suporte de Classe Única para discriminar diferentes condições de integridade, usando características sensíveis a danos extraídos da reconstrução de sinal do Autocodificador Variacional. Para comprovar a eficiência dessa metodologia, foram realizados ensaios em uma estrutura de aço e obtidos

resultados satisfatórios.

Pesquisas alinhadas ao presente trabalho

Conforme descrito anteriormente, pode-se constatar uma evolução natural do estudo de aprendizado de máquina mundialmente. Nos parágrafos seguintes, são elencados outros trabalhos que se fizeram essenciais para o desenvolvimento da presente dissertação.

Diversas são as pesquisas que demonstram que o VAE é um método que permite identificar com precisão os danos estruturais em uma edificação. Ma *et al.* [39] utilizaram o VAE para processar respostas de uma estrutura, reduzindo os dados de grande dimensão para um espaço de menor dimensão. Ainda nessa pesquisa, o VAE foi aplicado em uma ponte com um veículo em movimento, e tanto as simulações numéricas quanto os experimentos realizados produziram bons resultados.

Já no estudo dos autores Anaissi *et al.* [40] foi possível observar uma aplicação do VAE um pouco diferente. Nesse caso, a pesquisa utiliza o Autocodificador Variacional multi-objetivo para a detecção e o diagnóstico de danos de infraestrutura inteligente em dados de detecção multidirecional com base na probabilidade de reconstrução da rede neural. Assim, foi possível observar como se comporta quando se fundem dados de vários sensores com uma rede neural que utiliza recursos informativos para extrair informações que possam vir a identificar danos. Esse algoritmo gera pontuações probabilísticas de anomalias para detectar danos, avaliar a sua magnitude e localizá-lo. Nesse estudo, avaliou-se a metodologia através de um conjunto de dados obtidos em laboratório e também por meio de monitoramento de estruturas reais para fins de diagnósticos de danos. Após a coleta dessa base de dados, o algoritmo foi implementado em resultados experimentais tanto de laboratório, quanto em construções reais. Os resultados obtidos nesse estudo mostram que o método proposto pode detectar danos estruturais com precisão, estimar os diferentes níveis de gravidade da anomalia e identificar onde ela está sob um aspecto não supervisionado.

Terbuch [17] apresentou um sistema de aprendizado de máquina híbrido paralelo para a identificação de anomalias em grandes conjuntos de dados de

medição de séries temporais multivariadas. Nesse trabalho, com o uso de um Autocodificador Variacional de memória de curto prazo (LSTM-VAE), foi possível detectar o conjunto de dados anômalos. Como os resultados dessa pesquisa foram satisfatórios, foi extraído dali o algoritmo usado como parte central da presente dissertação, haja vista que o AE utilizado pode ser alterado e customizado de acordo com parâmetros que se entenda ser os mais adequados para cada caso analisado.

No artigo dos autores Terbuch *et al.* [41], foi investigado o uso de aprendizado de máquina híbrido para a detecção de séries temporais multivariadas. Para a detecção de anomalias nesse estudo híbrido, foram utilizadas técnicas com base em aproximadamente 50 indicadores de desempenho acopladas a um Autocodificador Variacional não-supervisionado com camadas de memória de curto prazo. Desse artigo foi retirado o código que otimiza os parâmetros da Rede Neural Artificial utilizada nesta pesquisa. Assim, adaptando-se os dois trabalhos dos autores Terbuch [17] e Terbuch *et al.* [41], foi possível criar o código da presente pesquisa que foi implementado no software *Matlab R2023a* ®. Para fins de divulgação científica, o código encontra-se descrito no Apêndice A (seção 6.1).

Em sua tese de doutorado, Finotti [1] apresentou a avaliação do algoritmo de aprendizado profundo denominado Autocodificador Esparso (SAE, do inglês *Sparse AutoEncoder*) quando utilizado como extrator de parâmetros de dados dinâmicos. A autora avaliou a capacidade dos modelos SAE de fornecer conjuntos de variáveis representativas, determinados através do processamento de sinais no domínio do tempo e que viabilizassem a detecção de um comportamento considerado normal para uma dada estrutura. Assim, foram analisadas quatro aplicações, sendo uma delas supervisionada e três não-supervisionadas. Como os resultados foram satisfatórios, além da base de dados dessa tese, foram extraídos também alguns conceitos que serão aplicados na presente dissertação.

Com os trabalhos anteriormente descritos, e apoiado sob o estudo e embasamento de outros, foi possível perceber que a aplicação do aprendizado profundo tem se tornado cada vez mais crescente. Assim, tendo em vista os resultados promissores obtidos até o momento, a presente pesquisa irá aplicar o VAE ao problema de SHM de estruturas civis, já que trata de algoritmos que podem prover uma maior

autonomia e precisão aos métodos de monitoramento estrutural fundamentados em inteligência artificial. As Tabelas 1 e 2 apresentam os principais estudos até aqui descritos de forma resumida.

Tabela 1 – Resumo dos principais estudos descritos - parte 1

Ano	Autor	Dados do estudo
1979	Cawley e Adams [21].	Pioneiros na avaliação da integridade estrutural utilizando a variação das frequências naturais como indicador de dano.
1992	Fox [22].	Apresentaram resultados numéricos e experimentais de uma viga fissurada utilizando das frequências naturais como indicador de dano.
1982	Allemang [24].	Detecção de danos usando as variações das amplitudes dos modos de vibração, através do MAC.
1982	Lieven e Ewins [25].	Detecção de danos usando as variações das amplitudes dos modos de vibração, através do COMAC.
1991	Pandey, Biswas e Samman [29].	Indicadores de danos baseados na variação da curvatura dos modos naturais de estruturas saudáveis e danificadas. Associando a mudança na rigidez de flexão à mudança na curvatura
1993	Kim e Stubbs [30].	Indicador evoluído baseado na variação da energia de deformação. Esses se mostraram aptos a detectar e localizar o dano em estruturas.
1994	Pandey e Biswas [31].	Localizou, detectou e quantificou o dano dado à redução de rigidez e ao aumento da flexibilidade da estrutura.
2011	Cury, Borges e Barbosa [33].	Estratégia que uniu a variação da energia de deformação e as frequências naturais para localizar e quantificar o dano, obtendo bons resultados.

Fonte: Autor.

Tabela 2 – Resumo dos principais estudos descritos - parte 1

Ano	Autor	Dados do estudo
2020	Ma *et al.* [37].	Autocodificador Variacional associado à uma aprendizagem não supervisionada. De modo a validar o estudo, aplicou o algoritmo em dados de uma ponte em funcionamento, obtendo bons resultados.
2020	Ma *et al.* [39].	Nesse estudo o VAE foi utilizado para processar respostas de uma estrutura, reduzindo a dimensionalidade dos dados. O método proposto foi aplicado a uma ponte com um veículo em movimento, obtendo bons resultados.
2021	Terbuch [17].	Sistema de aprendizado de máquina híbrido paralelo para a identificação de anomalias em grandes conjuntos de dados de medição de séries temporais multivariadas.
2022	Finotti [1].	Uso do Autocodificador Esparso (SAE) quando utilizado como extrator de parâmetros de dados dinâmicos. Foram analisadas quatro aplicações nesse estudo com resultados satisfatórios.
2023	Pollastro *et al.* [38].	Autocodificador Variacional associado à uma aprendizagem semi-supervisionada com uso de Máquina de Vetores Suporte de Classe Única para discriminar os danos. Para comprovar a eficiência dessa metodologia, foram realizados ensaios em uma estrutura de aço e obtidos resultados satisfatórios.
2023	Anaissi *et al.* [40].	Aplicou-se o Autocodificador Variacional multi-objetivo para a detecção e o diagnóstico de danos de infraestrutura inteligente em dados de detecção multidirecional com base na probabilidade de reconstrução da rede neural.
2023	Terbuch *et al.* [41].	Nesse estudo foi investigado o uso de aprendizado de máquina híbrido para a detecção de séries temporais multivariadas. Aqui usou-se o VAE em conjunto com a aprendizagem não-supervisionada com camadas de memória de curto prazo.

Fonte: Autor.

2.2 REFERENCIAL TEÓRICO

2.2.1 Aprendizado de Máquina

O uso de modelos preditivos através de algoritmos de inteligência computacional têm auxiliado cada vez mais no que tange à detecção de alterações estruturais. Desse modo, é essencial o estudo do aprendizado de máquina, o qual a sua essência está na capacidade de um sistema aprender com exemplos e experiência passadas, melhorando assim seu desempenho ao longo do tempo.

Nesse contexto, ao invés de seguir uma lógica programada, os algoritmos de aprendizado de máquina são concebidos para analisar grandes quantidades de dados e identificar padrões, correlações ou tendências entre eles. Assim, existem diferentes tipos de algoritmos, cada um ideal para um caso, dentre os quais podemos citar:

- **Aprendizado supervisionado:** o algoritmo é treinado usando os pares de saída e entrada conhecidos. Dessa forma, é possível mapear a correlação entre entradas e saídas desejadas com base nos exemplos fornecidos;
- **Aprendizado não supervisionado:** o algoritmo não recebe pares de entrada e saídas conhecidos. Desse modo, a máquina busca encontrar padrões nos dados, agrupando-os de acordo com sua semelhança. Uma vantagem desse método é a capacidade da criação automática de novos grupos ou classes [16];
- **Aprendizado semi-supervisionado:** nesse caso, uma parte dos dados é rotulada e outra parte não;
- **Aprendizado por reforço:** um agente é treinado para tomar sequências de decisões, recebendo recompensas ou penalidades em troca. O objetivo é que o agente aprenda a tomar as melhores ações para maximizar as recompensas ao longo do tempo. Esse tipo de aprendizado é amplamente aplicado na robótica e em jogos.

O presente trabalho utiliza de técnicas que englobam de aprendizado não supervisionado, a serem discutidas na próxima subseção.

2.2.2 Métodos de classificação

À medida que avançamos para um mundo cada vez mais orientado por dados, o aprendizado de máquina desempenha um papel crucial na análise e interpretação de dados. Desse modo, discorre-se a seguir um pouco mais sobre os métodos de classificação baseado em aprendizado de máquina utilizados neste trabalho.

Redes Neurais Artificiais

As RNAs têm sido extensamente empregadas na engenharia nos últimos anos em aplicações como: detecção de anomalias estruturais, classificadores de padrões, aprimoradores de sinal, previsão e controle do sistema, entre outros. No contexto de detecção de danos em estruturas, sua aplicação ocorre como um método de classificação de padrões, visto que tenta-se distinguir, a partir de dados de entrada, diferentes estados estruturais que caracterizam o sinal através da saída fornecida pela rede.

A classificação realizada por uma RNA é de caráter estatístico, onde as classes são representadas por pontos em um espaço de decisão multidimensional dividido em regiões que estão relacionadas a cada tipo de saída [42]. Assim, seus limites de decisões são tomados com base em um processo de treinamento e são concebidos tendo em vista a viabilidade estatística que existe entre as classes, através de uma de otimização realizada por meio da observação do erro de saída.

Segundo Braga *et al.* [16], as redes neurais são sistemas adaptativos formados por elementos de processamento (neurônios) que calculam algumas funções matemáticas que são geralmente não-lineares. Tais neurônios são distribuídos entre uma ou mais camadas e interligados através de conexões que são comumente unidirecionais. Assim, nesse modelo, as conexões estão associadas a certos parâmetros denominados pesos. Esses pesos são ajustados constantemente na fase de treinamento e, durante a fase de testes, possuem valores fixados. Além de possuir a função de armazenar o conhecimento visto no modelo, as redes fazem a ponderação de cada entrada recebida pelo neurônio daquela rede [43]. Após essa etapa, os neurônios somam todas as entradas ponderadas e produzem uma função não linear dessa soma, gerando uma saída [44].

Segundo Abraham [44], a larga utilização de RNAs para a solução de diversos tipos de problema com grande conjunto de dados não só se justifica pelo bom desempenho, como também pela otimização de tal tarefa. Como supracitado, primeiramente a rede neural passa por uma fase de aprendizado em que se é apresentado um dado problema. Posteriormente, essa metologia extrai as características essenciais da informação apresentada. Essa extração torna possível a solução de novos problemas com menor custo computacional [16], haja vista que generalizam (reduzindo as características) as informações aprendidas e gera saídas apropriadas para entradas nunca vistas antes [16] [43].

Nesse contexto, um dos tipos de RNAs mais empregados é o *perceptron* de múltiplas camadas (MLP, do inglês *Multilayer Perceptron*), o qual será um dos tipos utilizados no presente trabalho. O MLP é uma RNA do tipo *feedforward* que é um modelo que propaga as informações entre as conexões de neurônio de forma unidirecional, ou seja, da camada de entrada para a camada de saída. Além disso, tal rede neural é composta por uma camada de entrada, uma ou mais camadas ocultas (intermediárias) e uma de saída. Desse modo, os elementos de processamento (neurônios) trabalham com as funções de ativação [42], as quais possuem o objetivo de inserir uma não-linearidade ao modelo. Dessa forma, as MLP proporcionam propriedades analíticas mais complexas.

Dentre as inúmeras técnicas de aprendizado existentes para as MLP, a mais popular é a *backpropagation* [45]. Tal técnica de aprendizado supervisionado possui uma saída comparada com uma saída já conhecida, assim se obtém um erro com valor aceitável estipulado por uma função pré-definida ou um valor fixado. Posteriormente, o valor do erro obtido é propagado na camada de saída para as demais camadas, de modo a atualizar os pesos relacionados a cada conexão da MLP [45]. Assim, esse processo é repetido por um certo número de épocas (passagem completa por todo o conjunto de dados de treinamento) até que seja atendida uma métrica da diferença entre o valor das respostas alcançadas pela rede e a resposta de saída pré-fixada. Desta forma, ao término desse processo diz que a rede está treinada [43].

Realizar o mapeamento entre entradas e saídas nos pesos da rede é a base da aprendizagem de uma rede MLP [43]. Assim, para êxito desse mapeamento,

subdivide-se o conjunto de dados de entrada em três subconjuntos: treino, validação e teste. O primeiro é usado para treinar a rede, ou seja, ele é usado para que a rede neural possa adquirir informações que são utilizadas no processo de atualização dos pesos. Já as variáveis contidas no subconjunto de validação são usadas para a parada da fase de treinamento, que normalmente ocorre quando o erro obtido na validação ultrapassa o erro da etapa de treinamento. Após essa etapa, passa-se para a fase de teste, que basicamente se refere ao processo de execução da RNA já treinada.

A Figura 2 demonstra um exemplo de uma MLP com uma camada intermediária. A camada de entrada é definida pelos N parâmetros do espaço de entrada; a camada de saída é definida pelos M componentes do espaço de saída ou pela quantidade de classes existentes; e a camada oculta é definida empiricamente pelo número de neurônios K, os quais são responsáveis pela complexidade da superfície de separação da MLP [42].

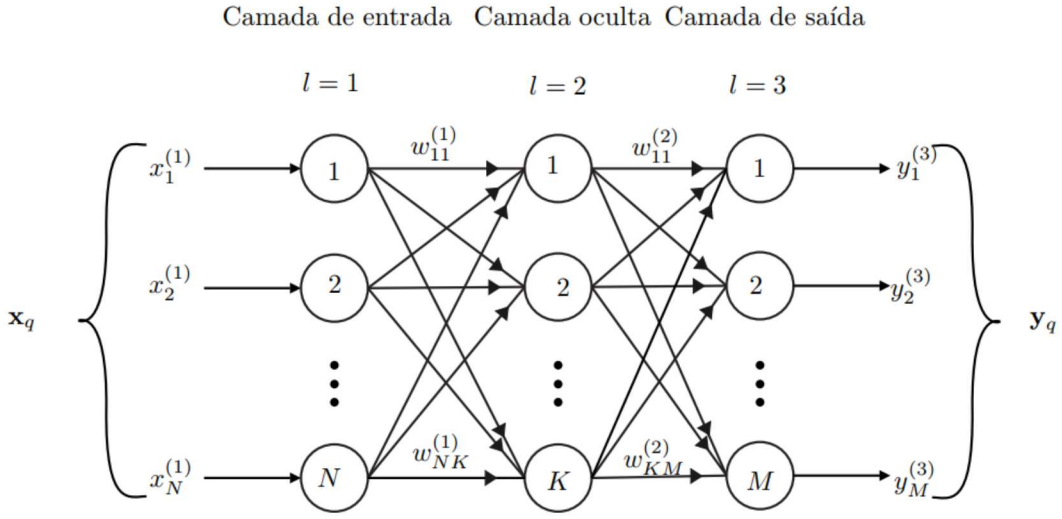


Figura 2 – Exemplo de uma MLP com uma camada oculta

Fonte: Finotti [1]

Desse modo, a informação é propagada da camada de entrada para a oculta e, por fim, para a camada de saída. Todo esse processo é realizado através das

funções de ativação. Assim, cada elemento de processamento da primeira camada oculta separa o espaço de entrada em regiões de decisão com base nos pesos. Por fim, a camada de saída possui a capacidade de combinar algumas dessas regiões por um efeito aditivo ou multiplicativo, de modo a gerar uma estrutura de separação cada vez mais complexa, segundo os ajustes realizados nos pesos das outras camadas [16] [42].

Os pesos presentes nas conexões das redes (representados por w) indicam a influência que cada entrada possui. Desse modo, baseado na MLP de três camadas apresentada na Figura 2, cada ensaio x_q a ser classificado possui N entradas (ou características) que são ponderadas pelos pesos w antes de serem processados pelos neurônios de cada camada. Assim, a soma ponderada de cada peso fornece um valor para a variável da função de ativação.

Nesse contexto, a caracterização das redes neurais artificiais é baseada em três aspectos fundamentais: a arquitetura da rede; as funções de ativação; e o algoritmo utilizado para a RNA.

Arquitetura

A arquitetura é importante na concepção de uma rede neural, haja vista que ela é quem limita os problemas a serem tratados pela rede [16]. Redes com apenas duas camadas (uma de entrada e outra de saída), resolvem apenas problemas mais simples, linearmente separáveis, tornando seu desempenho limitado [42]. Nesse sentido, para a melhoria desse desempenho, devem ser acrescentadas camadas intermediárias, as quais possibilitam a resolução de problemas não linearmente separáveis. Assim, os parâmetros que integram a arquitetura de uma rede neural são: o número de camadas existentes, o número de neurônios presentes em cada camada e a topologia da RNA [16].

Nesse sentido, redes que possuem uma configuração mais simples, como por exemplo uma camada oculta e um neurônio podem ser uma boa solução para problemas mais simples. No entanto, problemas mais complexos exigem cada vez mais camadas e mais neurônios. Segundo Bishop [45] e Cybenko [46], uma camada oculta é o bastante para aproximar funções contínuas e duas aproximam qualquer função matemática. No presente trabalho, utilizou-se uma ou duas camadas

intermediárias.

Nesse contexto, a definição adequada do número de elementos de processamento é uma das questões mais complexas do uso das RNAs [42]. Assim, há duas situações extremas: a da rede possuir camadas com poucos neurônios ou ter um número excessivo desses elementos. No primeiro caso, a RNA gastará um tempo excessivo na tentativa de encontrar uma solução adequada para o problema e pode não encontrá-la. No entanto, no segundo caso, os elementos processadores podem induzir a rede a memorizar os elementos de treinamento, prejudicando-a a generalizar para a fase de teste, na qual entram novos dados [16]. Desse modo, no que tange à escolha do número de neurônios a serem utilizados, em Braga *et al.* [16] há vários métodos para se realizar essa definição. Além disso, os autores ainda citam que para tal escolha alguns fatores influenciam como o número de dados de treinamento, quantidade de ruído existente nos dados, complexidade do problema a ser tratado e a divisão estatística dos exemplos de treinamento. Assim, normalmente o parâmetro em questão é definido empiricamente; após vários testes, a melhor configuração é definida. Na presente pesquisa, utilizou-se um método de otimização presente no trabalho dos autores Terbuch *et al.* [41], mas com intervalos de busca pré-estabelecidos de forma empírica.

Por fim, com relação às conexões existentes entre os neurônios, as RNAs podem ser do tipo *feedforward* ou *feedback*. Na primeira, como supracitado, a propagação do sinal ocorre de maneira unidirecional, pela camada de entrada e intermediária, até chegar nos neurônios da camada de saída. Já a segunda possui *loops*, permitindo que as informações sejam transmitidas de neurônios anteriores para neurônios futuros.

Ademais, a rede pode ser classificada também em relação às conexões entre seus neurônios, as quais podem ser divididas em totalmente conectadas ou parcialmente conectadas. A primeira, um neurônio de uma camada anterior se conecta a todos os neurônios da camada posterior (FC, do inglês, *fully conected*). Já a segunda, um neurônio da camada anterior não se conecta a todos os neurônios da camada posterior e são denominadas de arquiteturas de Redes Neurais Recorrentes (RNN) [43] [41]. Um exemplo bastante utilizado das RNNs é a LSTM (*Long Short-Term Memory*) em que a rede é projetada para lidar com problemas em que a

dependência temporal dos dados é crucial. Nesse tipo, há células de memórias que podem armazenar, acessar e remover informações ao longo do tempo, permitindo que a rede capture padrões de longo prazo. Outro tipo comum de arquitetura de RNN é a Bi-LSTM (Bidirectional Long Short-Term Memory). Essa é uma extensão da LSTM que processa a sequência de entrada tanto no sentido direto quanto no sentido reverso dos da rede. Desse modo, tal estrutura possui duas camadas de células LSTM de modo que uma processa os dados na sequência do sentido original (da entrada para a saída) e outra no sentido reverso (da saída para a entrada). Com isso, a Bi-LSTM auxilia na captura de padrões de dependências temporais em ambas as direções, o que pode ser útil onde a contextualização bidirecional é importante [41] [17]. No presente trabalho estes três tipos de arquiteturas foram avaliados.

Funções de ativação

Consideradas elementos chave na arquitetura de uma RNA, são responsáveis pela introdução da não-linearidade no processo de propagação de pesos da rede [43]. Nesse sentido, sem as funções de ativação, as redes seriam apenas modelos lineares, reduzindo sua capacidade de aprendizado e representação de padrões complexos nos dados.

Essas funções são aplicadas em cada neurônio da rede para determinar a saída do elemento processador com base em suas entradas ponderadas. Há vários tipos, sendo alguns exemplos listados a seguir:

- **Linear:** Usada no presente trabalho, essa função simplesmente retorna a soma ponderada das entradas sem aplicar nenhuma transformação não-linear. É usada principalmente em redes neurais lineares, onde a saída basicamente é uma combinação linear das entradas;
- **Degrau:** A função degrau atribui uma saída binária (0 ou 1) com base em um limite. Se o valor ponderado das entradas estiver acima do limite, o neurônio ativará e produzirá uma saída de 1, caso contrário, a saída será 0;
- **Sigmóide:** A função sigmóide mapeia os valores ponderados das entradas para um intervalo entre 0 e 1. Ela é amplamente utilizada em redes neurais

para modelar probabilidades ou realizar classificações binárias;

- **Tangente Hiperbólica:** A função tangente hiperbólica (tanh) mapeia os valores ponderados das entradas para um intervalo entre -1 e 1. Assim como a função sigmóide, a função tangente hiperbólica é comumente usada em redes neurais para introduzir não-linearidade.

Algoritmos de aprendizado

Há inúmeros algoritmos de aprendizado existentes para as RNAs. No presente trabalho, será dada ênfase apenas ao Autocodificador Variacional (VAE).

Um autoenconder é uma rede neural treinada por aprendizado não supervisionado, feita para aprender reconstruções que estão próximas de sua entrada original [20]. Uma rede neural autocodificadora possui um codificador e um decodificador, conforme mostram as Equações 2.1 e 2.2, respectivamente, onde W é o peso, b é o viés (ou *bias*) da rede neural e σ é uma função de transformação não-linear.

$$h = \sigma(W_{xh}x + b_{xh}) \tag{2.1}$$

$$z = \sigma(W_{hx}h + b_{hx}) \tag{2.2}$$

O codificador na Equação 2.1 mapeia um vetor x para uma representação em uma camada oculta h, seguindo uma não-linearidade. Já o decodificador na Equação 2.2 mapeia a representação oculta h de volta ao espaço de entrada original através de uma reconstrução realizada pela mesma transformação que o codificador. A diferença entre o vetor de entrada original x e a reconstrução z é chamada de erro de reconstrução, dada pela Equação 2.3. Um autocodificador aprende a minimizar esse erro de reconstrução. Dessa forma, o algoritmo para o autoenconder tradicional é mostrado abaixo (Tabela 3), onde f_θ e g_θ são redes neurais que possuem multicamadas.

$$\left\| x - z \right\| \tag{2.3}$$

Tabela 3 – **Algoritmo 1:** Algoritmo do Autocodificador

ENTRADA: Conjunto de dados $x^{(1)}, ..., x^{(N)}$
SAÍDA: Codificador f_θ, Decodificador g_θ
$\phi, \theta \leftarrow$ Parâmetros iniciais
Repetir
$E = \sum_{i=1}^{N} \left\|x^{(i)} - g_\theta(f_\theta(x^i))\right\|$ Calcula a soma do erro de reconstrução
$\phi, \theta \leftarrow$ Atualizar os parâmetros usando alguma métrica
até Convergir os parâmetros ϕ, θ

Fonte: An e Cho [20]

Nesse caso, os dados de treinamento são os dados normais. Após um treinamento eficiente, o Autocodificador tenderá a reconstruir os dados "normais" bem, diferentemente para o caso dos dados "anômalos". A Tabela 4 mostra o algoritmo de detecção de anomalias usando os erros de reconstrução dos Autocodificadores.

Tabela 4 – **Algoritmo 2:** Algoritmo de detecção de anomalia baseado em Autocodificadores

ENTRADA: Base de dados normal X, Base de dados anômalo $x^{(1)}, ..., x^{(N)}$, limite α
SAÍDA: Erro reconstruído $\|x - \hat{x}\|$ $\phi, \theta \leftarrow$ Treinam o Autocodificador utilizando X
Para $i = 1$ **até** N **faça**
Erro de reconstrução $(i) = \left\|x^{(i)} - g_\theta(f_\theta(x^i)\right\|$
Se *Erro de reconstrução* $(i) > \alpha$ **então**
$x^{(i)}$ é uma anomalia
Caso contrário
$x^{(i)}$ não é uma anomalia
Finaliza função se
Finaliza função para

Fonte: An e Cho [20]

Um VAE é um tipo de rede neural que combina modelos gráficos probabilísticos direcionados aos conceitos de Autocodificador, com o objetivo de aprender uma representação latente para os dados de entrada. Nesse sentido, além do que faz um Autocodificador tradicional - o qual busca apenas reconstruir a entrada original - o VAE também visa modelar a distribuição dos dados na forma de uma distribuição probabilística [20] [38].

Nesse estudo, utiliza-se o VAE para reduzir as dimensões das respostas estruturais de maneira probabilística. Assim, ainda segundo An e Cho [20] outra vantagem do uso desse tipo de Autocodificador é o fornecimento de medidas de probabilidade ao invés de um erro de reconstrução como forma de uma pontuação de anomalia, aqui denominada probabilidade de reconstrução. A ideia principal é tornar probabilístico tanto o codificador, quanto o decodificador. Desse modo, as variáveis latentes (z) são extraídas de uma distribuição de probabilidade que depende da entrada (X); já a reconstrução ($\hat{X}$), por sua vez, é escolhida probabilisticamente a partir de (z). A Figura 3 mostra a estrutura geral de um VAE.

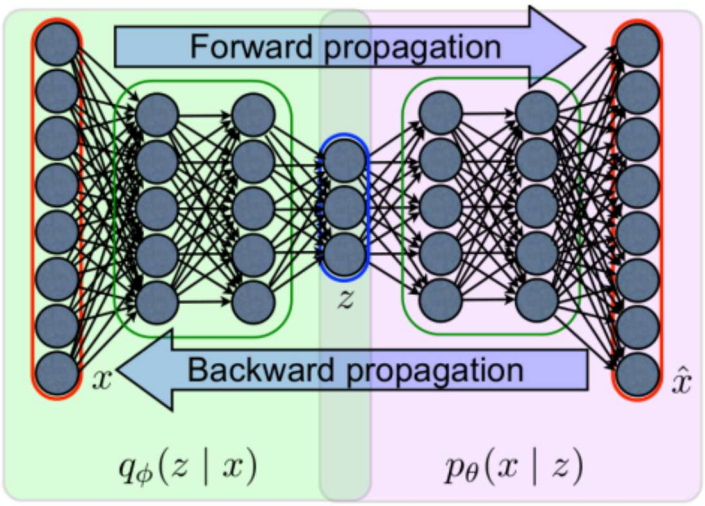


Figura 3 – VAE como codificador e decodificador

Fonte:An e Cho [20]

Inicialmente a RNA com essa arquitetura deve ser treinada. Para isso, o codificador recebe a entrada e retorna parâmetros para uma densidade probabilística (por exemplo, Gaussiana). Desse modo, a função de densidade probabilística para o codificador ($q_\phi(z|x)$) fornece as matrizes contendo a média e a covariância do modelo. Portanto, para cada entrada X, essa função fornece uma matriz de médias e uma diagonal ($\text{cov}(q_\phi(z|x))$), as quais determinam a densidade gaussiana. Nesse momento, os pesos ϕ precisam ser aprendidos pela RNA de modo que se consiga

configurar uma função de perda (a ser tratada mais adiante) e gerar o espaço latente.

Já o decodificador, recebe as variáveis latentes z e também retorna parâmetros para a densidade probabilística. Assim, a função de densidade probabilística para o decodificador $p_\theta(x|z)$ fornece as matrizes contendo a média e a covariância do modelo. De forma similar ao codificador, os pesos θ precisam ser aprendidos pela RNA de modo que se consiga configurar uma função de perda (a ser tratada mais adiante) e gerar o sinal reconstruído.

Antes de entender a função de perda para o VAE, é necessário ver alguns conceitos mais aprofundados. O primeiro deles é de que, para esse tipo de Autocodificador, para cada entrada x_i, maximiza-se o valor esperado do retorno de x_i; ou então minimiza-se a probabilidade esperada através da equação logarítmica negativa expressa na Equação 2.4.

$$-E_{q_\phi(z|x_i)}[log[p_\phi(x|z)]] \tag{2.4}$$

A Equação 2.4 leva o valor da variável latente z, sobre à distribuição atual de $q_\phi(z|x_i)$, à perda igual a $-log(p_\phi(x_i|z))$. Caso a função de perda fosse exatamente igual a Equação 2.4 nos casos em que a variável de entrada fosse as próprias variáveis latentes, as saídas poderiam ser muito diferentes. Para tentar solucionar esse problema, é necessário forçar $q_\phi(z|x_i)$ de tal modo que fique perto de uma distribuição normal (ou alguma outra distribuição de densidade simples). Para realizar esse feito, é introduzido nos cálculos um termo que considera a Divergência de Kullback-Leibler (ou Divergência KL).

A Divergência KL é uma medida que quantifica a diferença entre duas distribuições de probabilidade. Nesse caso, essa desigualdade será dada entre $q_\phi(z|x_i)$, que é a densidade probabilística para o codificador, e $p(z)$ uma distribuição fixa para qualquer entrada, o termo que representa o KL divergente é dado pela Equação 2.5.

$$D_{KL}(q_\phi(z|x_i)||p_\theta(z)) = \sum(q_\phi(z|x_i) \cdot log(\frac{q_\phi(z|x_i)}{p_\theta(z)})) \tag{2.5}$$

Desse modo, para um único conjunto de dados de entrada x_i, a função de

perda fica conforme a Equação 2.6.

$$l_i(\theta, \phi) = -E_{q_\phi(z|x_i)}[log[p_\phi(x|z)]] + D_{KL}(q_\phi(z|x_i)||p_\theta(z)) \tag{2.6}$$

O primeiro termo da Equação 2.6 promove a reconstrução dos sinais de entrada. Já o segundo termo, força que a codificação seja contínua, onde ela é comparada a um $p_\theta(z)$ fixo independentemente da entrada, o que inibe a memorização. Com a função de perda definida, o VAE pode ser treinado.

Após o treinamento, antes de validar ou testar como a RNA se comporta, é necessário fazer uma reparametrização, haja vista que a Equação 2.6 não é derivável em relação aos pesos θ ou ϕ. Para realizar essa operação, z é dado por $N(\mu(x_i), \sum(x_i))$. De modo a simplificar os cálculos, z pode ser reescrito como $z = \mu(x_i) + [\sqrt{\sum(x_i)}] \cdot \epsilon$, onde ϵ é $N(0, 1)$. Portanto, pode-se extrair amostras de $N(0, 1)$, que não dependem dos parâmetros.

Desse modo, após o treinamento, $q_\phi(z|x_i)$ está próximo de uma distribuição normal $N(0, 1)$. Além disso, ao ser utilizada uma amostra de z de $q_\phi(z|x_i)$, como entrada para $p_\theta(x|z)$, espera-se uma reconstrução aproximada de x_i. Por fim, caso se escolha qualquer parte z de $N(0, 1)$ e usá-la como entrada para $p_\theta(x|z)$, pode-se aproximar toda a distribuição de dados para $p(x)$. Assim, é possível gerar novas amostras que se parecem com a entrada, mas que não estão contidas nela.

Com o algoritmo treinado, realiza-se as fases de validação e teste. Nelas, como apresentado na Figura 4, a função $f(x, \phi)$ é uma rede neural que representa a relação entre os dados x e a variável latente z. Assim, para se obter z, com os dados de entrada x, a RNA aplica conceitos probabilísticos para definir a função $q_\phi(z|x)$ onde z é o produto da aplicação $q(z; f(x, \phi))$, reduzindo assim a dimensionalidade dos dados de entrada. Desse modo, para se obter a reconstrução ($\hat{x}$), dada a amostra z, a função de densidade probabilística, $p_\theta(x|z)$, é obtida por $g(z, \phi)$ onde $\hat{x}$ são amostradas por $p(x; g(z, \phi))$.

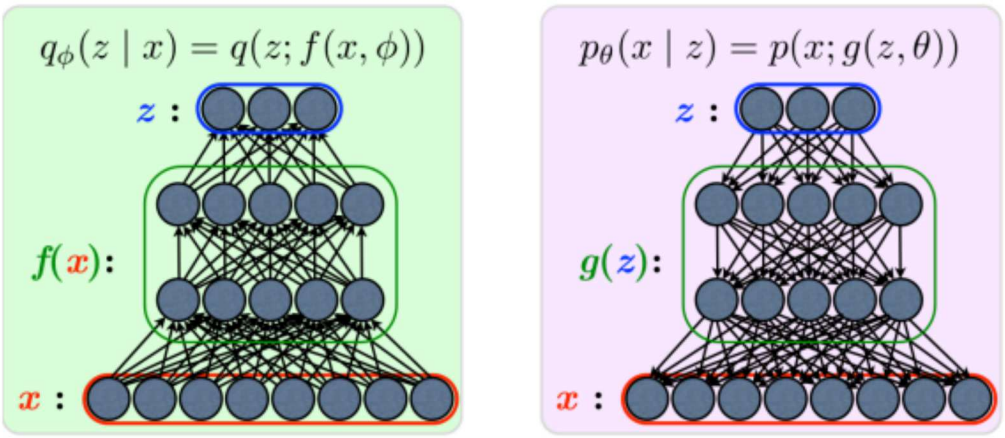


Figura 4 – Codificador e decodificador de um VAE

Fonte:An e Cho [20]

Em resumo, é a distribuição de parâmetros que está sendo modelada no VAE, não o valor em si. A escolha para as distribuições estão abertas a qualquer tipo de densidade probabilística. Assim, para as funções de densidade probabilística $p_\theta(z)$ e $q_\phi(z|x)$, a escolha comum é a distribuição normal, haja vista que assume-se que a relação entre as variáveis no espaço latente é muito mais simples do que nos dados de entrada original. Já para as distribuições probabilísticas $p_\theta(x|z)$, a escolha dependerá da natureza dos dados.

Métricas para avaliação dos erros de reconstrução dos sinais

Após reconstruir os dados de entrada, os erros entre o sinal de entrada e o de saída são extraídos como resultado. A análise desses erros possibilita o cálculo de métricas de distância, que, por sua vez, são empregadas para avaliar a proximidade ou dissimilaridade entre dois pontos em um espaço de características. Essas métricas desempenham um papel crucial em diversas tarefas, como agrupamento, classificação e detecção de anomalias.

Neste trabalho são utilizadas duas métricas de distâncias diferentes para se calcular os erros obtidos a partir da reconstrução dos sinais dinâmicos. A primeira delas é a função Erro Quadrático Médio (MSE, do inglês Mean Squared Error) apresentada na Equação 2.7, a qual mede a diferença entre o sinal de entrada e o

sinal reconstruído. Neste trabalho, a média dos MSE foi calculada para (n) ensaios, como mostra a Eq. 2.7 [47].

$$MSE = \frac{1}{n} \cdot [\sum_{n}^{i=1}(x_i - y_i)^2]^{\frac{1}{2}} \tag{2.7}$$

onde x_i representa o sinal original e y_i o sinal reconstruído.

A segunda métrica utilizada é uma combinação da MSE com a Distância de Mahalanobis [48]. Inicialmente, calcula-se o MSE de acordo com a Equação 2.7 para cada ensaio repetido com os mesmos níveis de danos (n). Posteriormente, calcula-se uma matriz V contendo o desvio-padrão dos resultados obtidos pelo MSE. Logo, V é calculada de acordo com a Equação 2.8:

$$V(n) = (MSE(n) - \overline{MSE(n)}) \tag{2.8}$$

onde $MSE(n)$ representa a matriz com os dados do Erro Quadrático Médio dos ensaios e $\overline{MSE(n)}$, a média.

Após esse passo, calcula-se a matriz de covariância de $\overline{MSE(n)}$, aqui chamada de S e expressa pela Equação 2.9:

$$S(n) = cov(\overline{MSE(n)}) \tag{2.9}$$

Por fim, calcula-se a Distância de Mahalanobis a partir da Equação 2.10:

$$D^2(n) = V' \cdot S^{-1} \cdot V \tag{2.10}$$

onde $D^2(n)$ representa a Distância de Mahalanobis.

No presente trabalho optou-se por determinar um limiar para discriminar os diferentes estados de dano de uma estrutura, fixado no valor referente a 95% dos resultados obtidos de erro de reconstrução por ensaio durante a fase de treinamento do algoritmo. No próximo capítulo esta abordagem será explicada mais detalhadamente.

3 METODOLOGIA

Conforme mencionado no Capítulo 1, o principal objetivo do presente trabalho é validar a eficiência do Autocodificador Variacional na detecção de danos estruturais. Para tanto, foi utilizado o código disponibilizado por Terbuch [17] que trata do método implementado, além de um outro, oriundo de um trabalho mais recente dos autores Terbuch *et al.* [41] que possibilita otimizar algumas das variáveis que compõem a arquitetura da Rede Neural Artificial. O código customizado utilizado nesta pesquisa foi implementado no *Matlab R2023a* ® e encontra-se disponível no Apêndice A (seção 6.1).

Resumidamente, a pesquisa adotou a seguinte metodologia:

1. Primeiramente, foram definidas duas bases de dados para aplicação do método desenvolvido. A fim de testar o algoritmo em um ambiente mais controlado, as respostas dinâmicas obtidas a partir de ensaios em laboratório de um pórtico bi-engastado foram utilizadas como primeira aplicação (Finotti *et al.* [49]). Posteriormente, essa metodologia foi aplicada ao dados da clássica ponte Z24, cuja base de dados encontra-se presente no trabalho dos autores Roeck *et al.* [50];

2. Em seguida, o algoritmo foi desenvolvido adaptando-se os códigos presentes nas pesquisas de Terbuch [17] e Terbuch *et al.* [41] com o objetivo de detectar possíveis anomalias em estruturas;

3. Para cada uma das aplicações, o modelo foi otimizado, treinado e testado;

4. Calcula-se o erro médio entre os sinais originais e reconstruídos pelo método para cada ensaio das bases de dados utilizadas;

5. Assim, com os erros médios obtidos, calcula-se os valores das métricas MSE e MSE com Mahalanobis (explicadas no item 2.2.2) a fim de agrupar os cenários de danos;

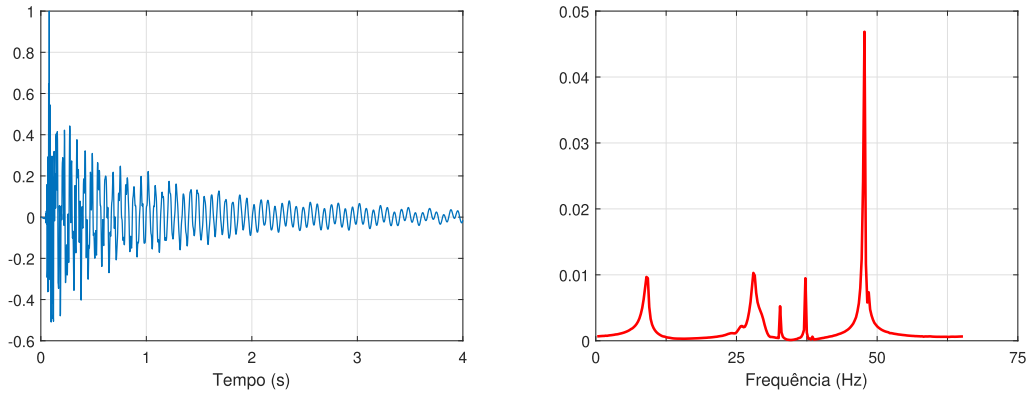
6. De modo a classificar os danos com base nos valores resultantes do passo anterior, define-se um limiar fixado acima de 95% dos resultados obtidos durante a fase de treinamento do algoritmo;

7. Assim, faz-se a classificação dos danos: se o valor obtido no passo 7 estiver abaixo do limiar, o ensaio é considerado sem dano; caso contrário, com dano.

A seção a seguir detalha o algoritmo usado neste trabalho.

3.1 ALGORITMO

Antes de iniciar o uso do algoritmo do VAE, em si, é necessário que se defina os dados que serão utilizados. Mais especificamente, o domínio de análise. No presente trabalho, as respostas dinâmicas são usadas como dados de entrada para o VAE representadas ora no domínio do tempo, ora no domínio da frequência. Para esta última, a Transformada Rápida de Fourier (FFT) foi empregada. A Figura 5(a) demonstra um sinal no tempo para a aplicação do pórtico plano, enquanto a Figura 5(b) mostra a mesma resposta, mas no domínio da frequência. É importante destacar que todos os sinais do domínio do tempo foram normalizados tomando-se como base o módulo da amplitude máxima de cada um. Esse processo gera resultados adimensionais no intervalo entre -1 e 1.



(a) No domínio do tempo (b) No domínio da frequência

Figura 5 – Resposta dinâmica típica do pórtico.

Fonte: Autor

Uma vez definida a estrutura dos dados de entrada, faz-se necessário definir alguns parâmetros intrínsecos à metodologia utilizada, a saber:

1. **Camadas do codificador:** Podem ser escolhidas três formas distintas de como os neurônios do codificador são conectados: FC, LSTM e Bi-LSTM. De maneira geral, usa-se camadas FCs em problemas de classificação ou regressão mais simples, onde não há uma dependência temporal intrínseca. Já em problemas de sequências temporais, usa-se camadas LSTMs para capturar essas dependências de longo prazo. Por fim, utiliza-se camadas Bi-LSTMs quando a informação tanto do passado quanto do futuro for relevante para a tarefa em questão. Além disso, para formar uma arquitetura mais profunda, é possível aumentar o número de camadas do codificador, repetindo o tipo de ligação ou até mesmo misturando-as. É importante salientar que o custo computacional cresce à medida que são adicionadas mais camadas. Além disso, a execução do método aumenta progressivamente ao utilizar camadas com FC, LSTM e Bi-LSTM, seguindo essa sequência;

2. **Camadas do decodificador:** Igual ao explicado para o codificador;

3. **Neurônios do codificador:** Aqui devem ser especificados os números de neurônios para cada camada definida para os codificadores. Após os testes, foi visto que uma boa estimativa inicial corresponde à metade do valor de características da camada anterior. Ademais, cabe ressaltar que quanto maior esse número, maior o custo computacional. Ademais, não necessariamente quanto maior o número de neurônios, melhor será a representação dos dados de saída;

4. **Neurônios do decodificador:** Igual ao explicado para o codificador;

5. **Número de variáveis na camada latente:** Especifica o número de variáveis latentes. Essas variáveis são responsáveis por construir uma representação compacta dos dados de entrada. Nesse caso, deve ser colocado um número menor que o número de características dos dados de entrada. Bem como para os neurônios, quanto maior o número desse parâmetro, maior o tempo computacional. Novamente, não necessariamente quanto maior o número de neurônios, melhor será a representação dos dados de saída;

6. **Número de épocas:** Define o número de épocas realizadas durante a fase de treinamento. Aqui é especificado o número de ciclos em que a rede passa

por todos os dados de treinamento para atualizar os seus parâmetros. Quanto maior o número de épocas, maior será o tempo computacional. O melhor valor para o número de épocas depende de cada caso;

7. **Número de características:** Define o número de atributos que os dados de entrada possuem. Esse valor é extraído automaticamente;

8. **Taxa de aprendizado:** É um valor que determina a magnitude com a qual os pesos da rede são ajustados durante o processo de treinamento. Ou seja, a taxa de aprendizado controla o tamanho dos passos que a otimização dá ao ajustar os pesos da rede para minimizar a função de custo ou perda. Em termos gerais, quanto menor a taxa de aprendizado, melhor tenderá a ser a otimização dos parâmetros e maior será o custo computacional;

9. **Tamanho do minilote:** Durante cada iteração, apenas uma pequena amostra (minilote) de todo o conjunto de dados é usada. Logo, nesse parâmetro deve ser informado qual o tamanho do minilote. Normalmente, quanto menor, melhor o treinamento da rede e maior o tempo computacional;

10. **Local de execução do treinamento da rede neural:** Aqui deve ser indicado se a execução da fase de treinamento da RNA será feita em uma CPU ou GPU, caso a máquina usada tenha placa de vídeo. Há também a possibilidade de se deixar esse parâmetro automático, indicando apenas 'AUTO' para o algoritmo;

11. **Divergência de Kullback-Leiber:** O código solicita que esse parâmetro de ponderação seja indicado. Como dito anteriormente, A Divergência de Kullback-Leibler (KL, Equação 2.5) é uma medida da diferença entre duas distribuições de probabilidade. Ela é usada para avaliar o quão bem essas distribuições estão próximas. Normalmente, quanto menor o valor de KL, melhor;

12. **Função de saída dos dados:** Refere-se à camada final da rede que produz os resultados ou previsões desejadas com base nos dados de entrada. Caso o usuário opte por saídas entre 0 e 1, deve-se colocar 'sigmoid' para esse parâmetro. Para saídas entre -1 e 1, 'tanh' deve ser fornecido. Se não são

desejadas nenhuma função de saída, basta colocar 'none'. Se a tarefa for uma regressão linear, a função de saída mais adequada serão os próprios valores reconstruídos pela rede, logo o usuário deve informar 'none'. Para problemas de classificação binária, onde a saída é uma escolha entre duas classes, a função de saída deve ser a sigmoide. Para problemas de classificação com mais de duas classes a função ideal a ser escolhida é a 'tanh';

Em seguida, mediante a combinação da recente metodologia incorporada a um algoritmo genético apresentado pelos autores Terbuch *et al.* [41], tornou-se viável a otimização de alguns dos parâmetros mencionados anteriormente, requerendo-se apenas a inserção por parte do usuário dos seguintes parâmetros:

1. **Número de populações:** Após vários testes empíricos, o valor desse critério foi igual a 20, o qual melhor se ajustou aos dados testados;
2. **Número de gerações:** Número de vezes que o código deverá ser executado caso não convirja antes. Essa visualização da convergência é dada através da diferença entre o melhor ajuste possível da rede e os seus ajustes médios. Após vários testes empíricos, o valor desse critério foi igual a 10;
3. **Intervalo de parâmetros:** Nessa fase, devem ser indicados os intervalos dos parâmetros a serem otimizados (4, 5, 7, 9 e 12 da lista anterior). Assim, o código retornará a melhor configuração deles para realizar aquela tarefa.

Com a arquitetura definida e otimizada, a próxima etapa é a de tratamento de dados para a classificação. Nesse passo, o código começará a processar os dados e retornará uma saída para cada dado de entrada. Como citado anteriormente, nesse trabalho foram usadas duas métricas de distâncias diferentes para calcular o erro entre os dados de saída e entrada para cada nível de dano.

Por fim, o último passo é a classificação dos dados. Para essa etapa foi calculada uma reta (limiar) acima de 95% dos valores de erro obtidos na fase de treinamento e realizada a classificação dos sinais. A Figura 6 mostra um resumo de como funciona a metodologia proposta.

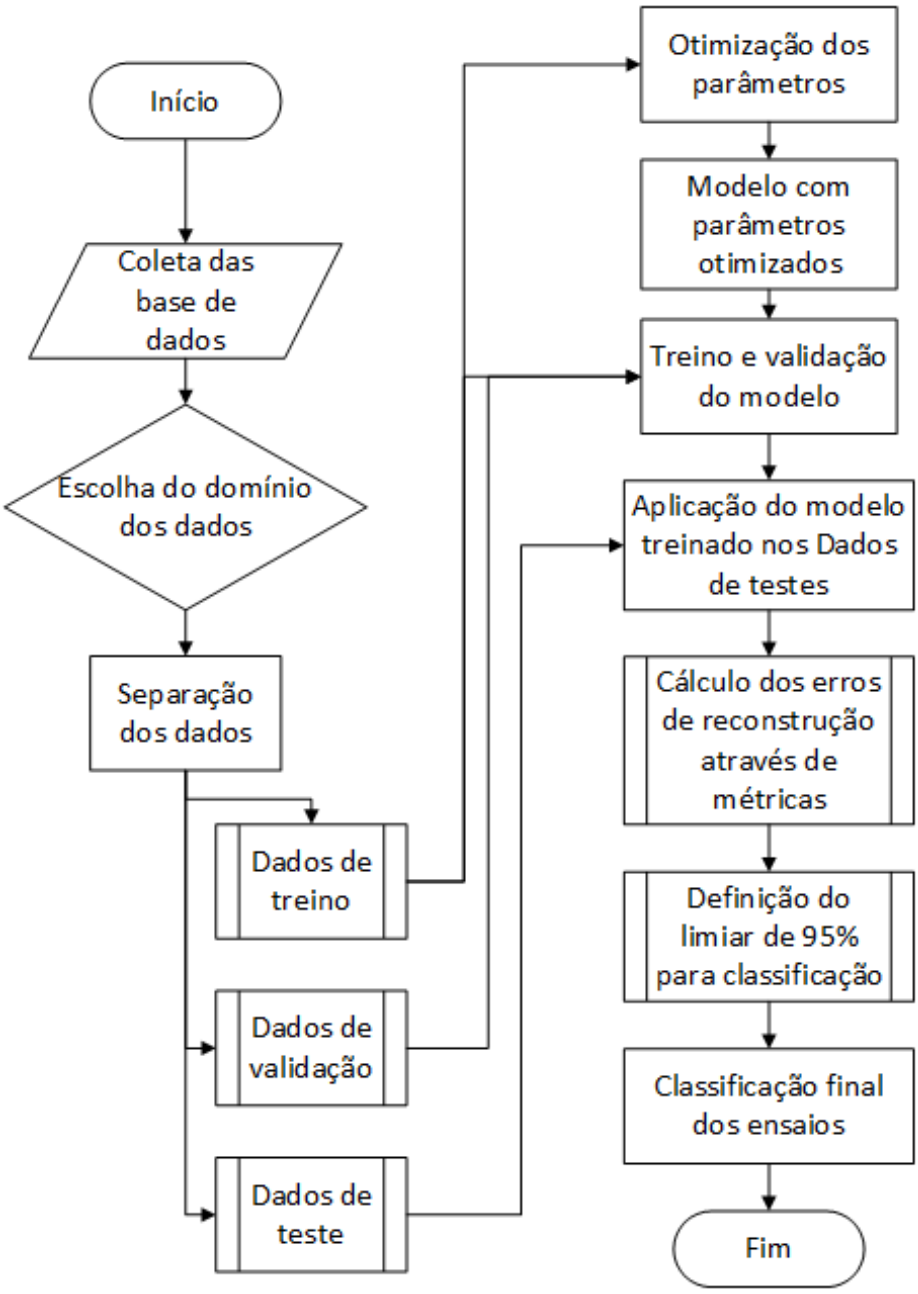


Figura 6 – Fluxograma do método proposto

Fonte: Autor

No capítulo seguinte serão descritas, em detalhes, as aplicações utilizadas neste trabalho para a validação da metodologia proposta.

4 APLICAÇÕES E RESULTADOS

4.1 PÓRTICO BI-ENGASTADO

A primeira aplicação utilizada para validar a estratégia proposta é mostrada Figura 7. Trata-se de um pórtico de alumínio montado e instrumentado no Laboratório de Imagens e Sinais (LIS) da Universidade Federal de Juiz de Fora [49]. A estrutura é formada por 6 barras de alumínio de 300mm de comprimento × 15,875mm de largura e 1,587mm de espessura.

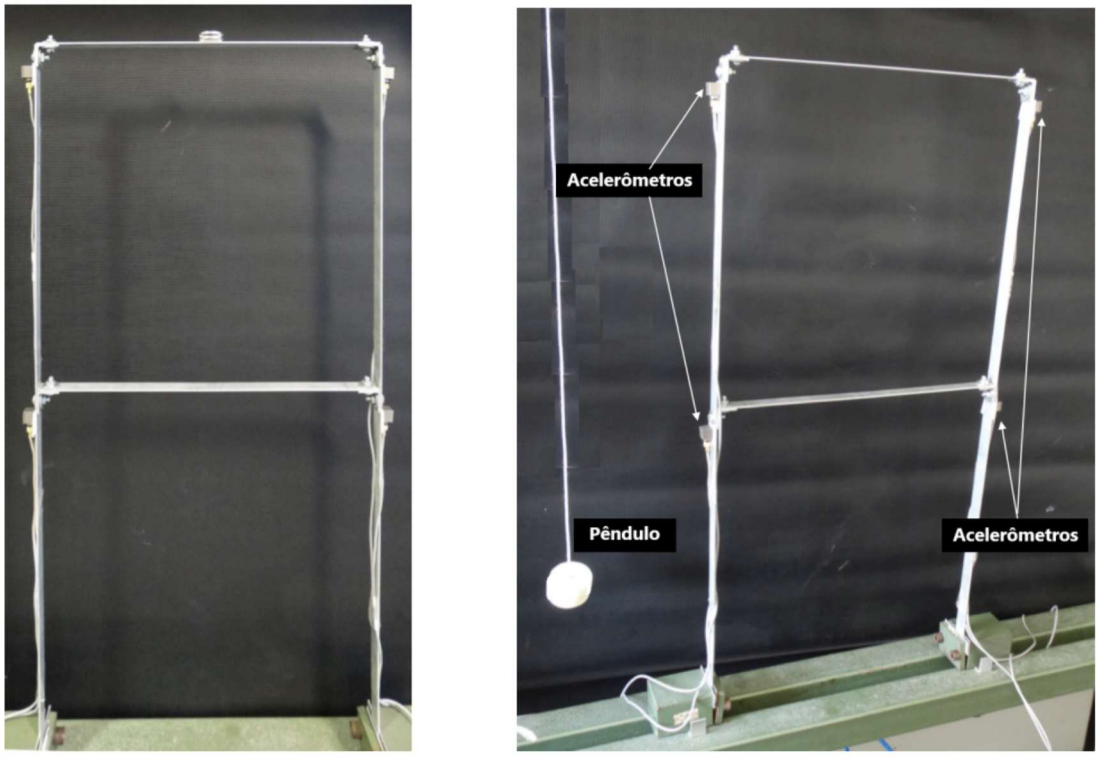


Figura 7 – Pórtico ensaiado

Fonte:Finotti [1]

Segundo Finotti *et al.* [49], os ensaios dinâmicos dispuseram de quatro acelerômetros piezoelétricos, responsáveis por medir as respostas estruturais causadas pelo impacto da carga aplicada por meio de um pêndulo de 14 g, conforme apre-

sentado na Figura 8. Desse modo, partindo do repouso, a massa era abandonada sempre de uma altura fixa até colidir com o pórtico. Além disso, de modo a alterar o comportamento dinâmico da estrutura ao longo do ensaio, as massas m_1 e m_2 foram afixadas de forma gradual às barras do pórtico. Durante os ensaios, foram consideradas cinco configurações diferentes, a saber:

- Cenário 1 - Nenhuma massa adicional afixada à estrutura: m_1 e m_2 iguais a zero. Esse cenário é denominado “dano 0”;
- Cenário 2 - Uma massa adicional afixada à estrutura: $m_1 = 7{,}81$g e $m_2 = 0$. Esse cenário é denominado “dano 1”;
- Cenário 3 - Uma massa adicional afixada à estrutura: $m_1 = 15{,}62$g e $m_2 = 0$. Esse cenário é denominado “dano 2”;
- Cenário 4 - Duas massas adicionais afixada à estrutura: $m_1 = 15{,}62$g e $m_2 = 7{,}81$g. Esse cenário é denominado “dano 3”;
- Cenário 5 - Duas massas adicionais afixada à estrutura: $m_1 = 15{,}62$g e $m_2 = 15{,}62$g. Esse cenário é denominado “dano 4”;

Os dados analisados para cada cenário de dano incluem 300 ensaios no total. Logo, são 300 sinais diferentes por acelerômetro, todos registrados a uma frequência de 500 Hz durante 8,182s. Portanto, 4096 pontos por sinal. Desse modo, um resultado típico desse teste, tanto no domínio do tempo como no da frequência, é mostrado nas Figuras 9(a) e 9(b), respectivamente. Em ambas as figuras, apenas 2000 pontos por sinal são apresentados. É importante destacar que as amplitudes dos sinais são adimensionais, haja vista a normalização explicada no Cap. 3. No contexto do domínio da frequência (Figura 9(b)), mesmo com o sistema possuindo apenas 2 graus de liberdade, durante o ensaio, a estrutura foi excitada de tal forma que mais de duas frequência naturais pudessem ser identificadas.

Nesse contexto, os testes foram divididos em três estudos diferentes que se diferenciam de acordo com a arquitetura da rede adotada.

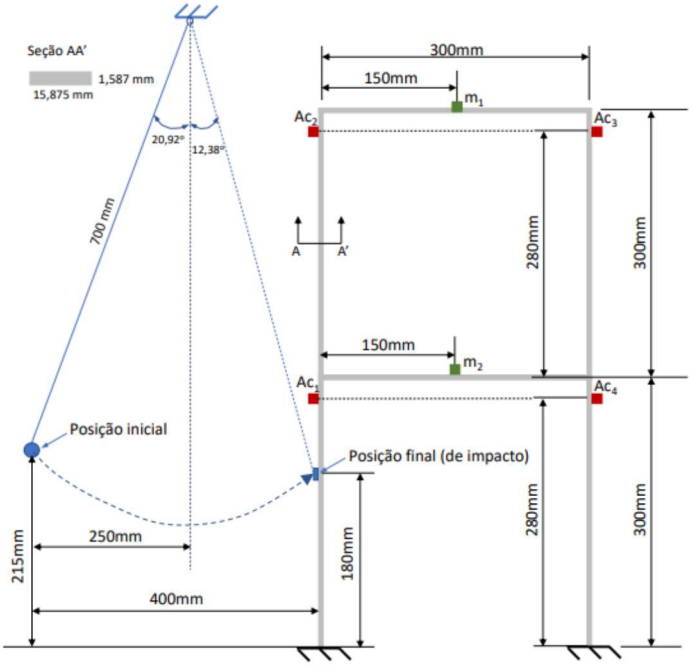
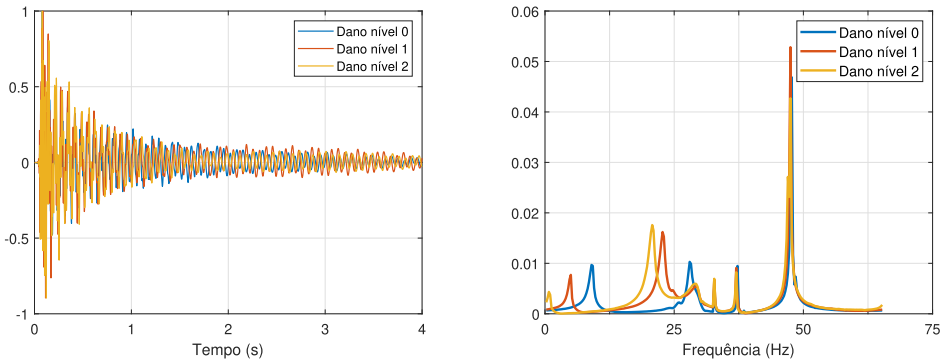


Figura 8 – Esquema experimental do pórtico.

Fonte: Finotti [1].



(a) No domínio do tempo (b) No domínio da frequência

Figura 9 – Respostas dinâmicas típicas do pórtico ilustrando 3 níveis de danos.

Fonte: Autor

4.1.1 Arquitetura das redes

De modo a analisar o maior número de redes possíveis durante o período dessa pesquisa, foram testados vários modelos, conforme apresentado no Apêndice B (Seção 6.2). Inicialmente, foram variados de maneira empírica: os tipos de ligações entre as camadas de codificação e decodificação (alternando-se entre FC, LSTM e Bi-LSTM); o número de neurônios presentes nessas camadas; a dimensão do espaço latente e o número de épocas do algoritmo. Nessa etapa, foram utilizados apenas os três primeiros cenários de dano e realizados pouco mais de 1000 testes com duração aproximada de 15 minutos cada, em um computador com as seguintes configurações: Intel Core i7-6700K CPU 2 núcleos 4.00 GHz e 16 GB de RAM. Nessa fase preliminar, foi possível perceber a capacidade do VAE para a detecção de danos na estrutura.

Durante a segunda etapa, com o uso de parte do código dos autores Terbuch *et al.* [41], foi implementada a otimização de boa parte desses parâmetros e testados os cinco cenários de danos. Além disso, foram variados novos números de camadas latentes (já que o algoritmo original não otimiza essa parte) e novas funções erros (MSE, MSE com a Distância de Mahalanobis, ORSR e Resíduo). Nesta etapa, o modelo retornava resultados melhores, mas levava mais tempo para ser executado (variando de 4 a 10 horas, para cada ciclo). Em razão disso, foram realizados 240 testes, otimizando-se apenas o modelo relativo ao acelerômetro 3 e aplicando-se os mesmos parâmetros otimizados aos demais. Nessa etapa, foi possível observar que as métricas de erros ideais dentre as testadas foram a MSE e a MSE+Mahalanobis. Além disso, foi visto que o melhor número de neurônios nas camadas latentes era 75 e 100. Todos os demais parâmetros de otimização foram mantidos, com exceção do intervalo de neurônios e as funções presentes na camadas codificadoras e decodificadores, até então do tipo FC.

Os testes conduzidos nas fases anteriores foram restritos aos dados de entrada no domínio do tempo. Portanto, a etapa subsequente envolveu a variação do domínio dessas bases, abrangendo tanto o domínio da frequência quanto a integração do tempo com a frequência. Ademais, os dados estavam sendo reduzidos de maneira brusca (de 2.000 por entrada para 100 na primeira camada) e os decodificando da mesma forma. Então, para melhorar essa etapa mudou-se o

intervalo de otimização dos neurônios das camadas codificadora e decodificadora. Após rodar 14 modelos distintos para o acelerômetro 3 (que variam em seu tempo de execução de 6 a 12 horas), foi possível identificar o intervalo de números de neurônios mais adequado para otimização. Além disso, concluiu-se que o domínio com que os dados de entrada são inseridos afetam o resultado final dos erros de reconstrução. Após essa etapa, utilizando os parâmetros otimizados, o modelo foi testado nos demais acelerômetros para cada um dos 14 modelos.

Assim, para cada tipo de ligação diferente entre os neurônios (FC, LSTM e Bi-LSTM), foi escolhida a melhor arquitetura (modelos A, B e C). Em todas as simulações, a matriz de entrada no algoritmo é sempre a mesma: para cada acelerômetro, por cenário, são 300 ensaios contendo 2000 pontos (o sinal foi cortado a partir do momento em que a amplitude das respostas dinâmicas passam a ser desprezíveis). Desse modo, após a otimização do modelo para cada acelerômetro, todos os sinais de entrada passam pelo seguinte processo:

- **Fase de treinamento:** Inicialmente, o modelo é treinado utilizando os dados do cenário de dano 0. Assim, 200 ensaios foram separados para o treinamento da rede e 100 para validação;
- **Fase de teste:** Posteriormente, o modelo é testado com os 300 ensaios relativos a cada um dos outros cenários de danos (de 1 a 4).

O que diferem as redes testadas são os tipos de ligações entre os seus neurônios. A arquitetura A possui todos os neurônios conectados com a camada subsequente, ou seja, do tipo FC. Já a arquitetura B, possui células de memórias que podem armazenar, acessar e remover informações ao longo do tempo, permitindo que a rede capture padrões de longo prazo (ou seja, do tipo LSTM). Por fim, o tipo de ligação entre os neurônios da arquitetura C é a Bi-LSTM, a qual possui duas camadas de células LSTM, de modo que uma processa os dados na sequência do sentido original e outra no sentido reverso.

Os parâmetros utilizados no algoritmo são:

1. **Tipo de Autocodificador:** 'VAE';

2. **Camadas do codificador:** Para todas as simulações, foi utilizada apenas uma camada. Para o modelo A essa camada é a FC, para o B a LSTM e para o C a Bi-LSTM;

3. **Camadas do decodificador:** Igual ao codificador;

4. **Neurônios do codificador:** O intervalo escolhido para otimização foi entre 1000 e 1100 neurônios. Esse intervalo é definido como sendo aproximadamente a metade do número de características dos dados de entrada;

5. **Neurônios do decodificador:** Igual ao codificador;

6. **Número de variáveis na camada latente:** Após várias simulações, os melhores resultados são: para o modelo A igual a 75 e, para os modelos B e C, 100;

7. **Número de épocas:** O intervalo escolhido para otimização foi entre 0 e 100 épocas;

8. **Número de características:** Número retornado pelo programa e igual a 2000. Esse número corresponde ao tamanho do vetor do comportamento dinâmico de um ensaio;

9. **Taxa de aprendizado:** O intervalo escolhido para otimização foi entre 0,00001 e 0,001;

10. **Tamanho do minilote:** O intervalo escolhido para otimização foi entre 2 e 100 para esse parâmetro;

11. **Local de execução do treinamento da rede neural:** Os computadores que foram utilizados nos testes não possuíam placa de vídeo. Nesse caso, foi colocado 'AUTO' nos modelos, mas poderia ser 'CPU';

12. **Divergência de Kullback-Leiber:** O intervalo escolhido para otimização foi entre 0 e 100;

13. **Função de saída dos dados:** Não se utilizou nenhuma função de saída para os modelos. Portanto, esse parâmetro foi fixado em 'none';

Assim, são inseridos os dados na rede neural com o Autocodificador Variacional e o algoritmo retorna um sinal reconstruído. Após essa etapa, é calculado o erro entre o sinal reconstruído e o original para cada ensaio através de uma métrica. Para as redes A e C a melhor métrica definida foi a MSE, já para a rede B a métrica adotada foi a combinação da MSE com a Distância de Mahalanobis.

Nesse sentido, a etapa seguinte consiste na classificação desses erros, com a finalidade de conhecer a capacidade do algoritmo de distinguir os cenários de danos. Como dito anteriormente, para esse trabalho foi utilizado o limiar de 95% dos erros médios obtidos da fase de treinamento do código para classificação.

4.1.2 Resultados obtidos

Para esta aplicação, apesar de terem sido testados três domínios de análise diferentes (tempo, frequência e uma combinação de dados nesses dois domínios), todos os melhores modelos foram obtidos para os dados no domínio do tempo.

Assim, iniciada a fase de otimização da arquitetura da rede, o algoritmo retorna como primeiro resultado o gráfico mostrado na Figura 10. Nele, o usuário pode visualizar a diferença entre o melhor ajuste possível dos parâmetros da rede e os ajustes médios. Desse modo, vê-se a necessidade de se aumentar o número de gerações ou o número de indivíduos para aproximar cada vez mais esses valores. Portanto, para cada otimização realizada por acelerômetro, foi avaliado se o número de população (definido como 20) e o número de gerações (fixado em 10) estão adequados. Para todos as simulações, esses valores foram suficientes.

Após essa etapa, o algoritmo também retorna os parâmetros da rede otimizada. As Figuras 11, 12 e 13 mostram os resultados obtidos por acelerômetro.

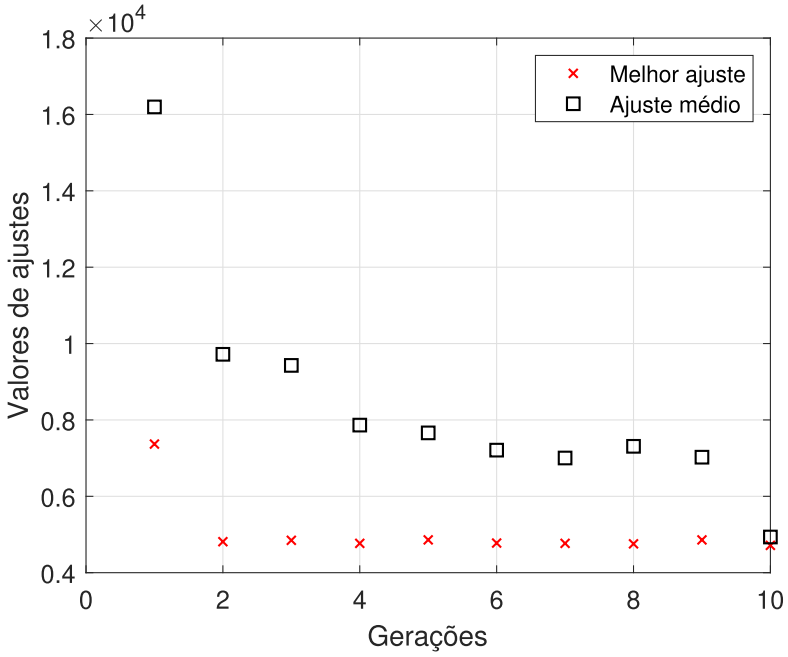


Figura 10 – Exemplo de um gráfico de gerações do modelo A - Acelerômetro 1.

Fonte: Autor

OTIMIZAÇÃO - PÓRTICO - MODELO A				
Parâmetro	**AC1**	**AC2**	**AC3**	**AC4**
AutoencoderType	VAE	VAE	VAE	VAE
LayersEncoder	FC	FC	FC	FC
LayersDecoder	FC	FC	FC	FC
NeuronsEncoder	1066	1050	1072	1046
NeuronsDecoder	1047	1083	1050	1056
LatentDim	75	75	75	75
NumberEpoch	38	51	56	75
NumberFeature	2000	2000	2000	2000
LearningRate	0,0004915	0,0006	0,0005949	0,0005801
MiniBatchSize	83	84	81	81
ExecutionEnvironment	AUTO	AUTO	AUTO	AUTO
WeightingKL	25	12	6	3
OutputTransferFunction	NONE	NONE	NONE	NONE
Domínio	TEMPO	TEMPO	TEMPO	TEMPO
Classificação	MSE	MSE	MSE	MSE

Figura 11 – Parâmetros otimizados, por acelerômetro, Modelo A. Fonte: Autor.

OTIMIZAÇÃO - PÓRTICO - MODELO B				
Parâmetro	**AC1**	**AC2**	**AC3**	**AC4**
AutoencoderType	VAE	VAE	VAE	VAE
LayersEncoder	LSTM	LSTM	LSTM	LSTM
LayersDecoder	LSTM	LSTM	LSTM	LSTM
NeuronsEncoder	1055	1071	1055	1055
NeuronsDecoder	1042	1031	1042	1042
LatentDim	100	100	100	100
NumberEpoch	95	88	95	95
NumberFeature	2000	2000	2000	2000
LearningRate	0,0008129	0,0007378	0,0007991	0,0006965
MiniBatchSize	81	70	15	55
ExecutionEnvironment	AUTO	AUTO	AUTO	AUTO
WeightingKL	5	0	72	5
OutputTransferFunction	NONE	NONE	NONE	NONE
Domínio	TEMPO	TEMPO	TEMPO	TEMPO
Classificação	MSE+MAHALA	MSE+MAHALA	MSE+MAHALA	

Figura 12 – Parâmetros otimizados, por acelerômetro, Modelo B

Fonte: Autor

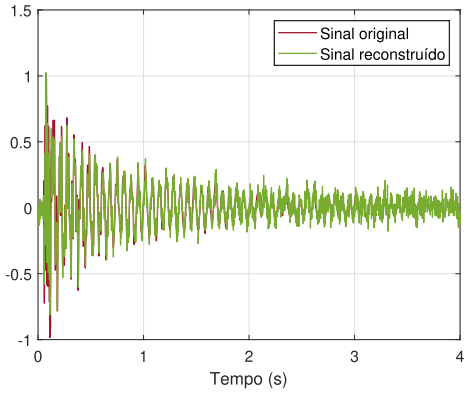
OTIMIZAÇÃO - PÓRTICO - MODELO C				
Parâmetro	**AC1**	**AC2**	**AC3**	**AC4**
AutoencoderType	VAE	VAE	VAE	VAE
LayersEncoder	Bi-LSTM	Bi-LSTM	Bi-LSTM	Bi-LSTM
LayersDecoder	Bi-LSTM	Bi-LSTM	Bi-LSTM	Bi-LSTM
NeuronsEncoder	1087	1055	1010	1065
NeuronsDecoder	1070	1051	1060	1091
LatentDim	100	100	100	100
NumberEpoch	82	95	94	82
NumberFeature	2000	2000	2000	2000
LearningRate	0,0006648	0,0007405	0,0008939	0,000818
MiniBatchSize	41	28	60	79
ExecutionEnvironment	AUTO	AUTO	AUTO	AUTO
WeightingKL	19	5	14	14
OutputTransferFunction	NONE	NONE	NONE	NONE
Domínio	TEMPO	TEMPO	TEMPO	TEMPO
Classificação	MSE	MSE	MSE	MSE

Figura 13 – Parâmetros otimizados, por acelerômetro, Modelo C

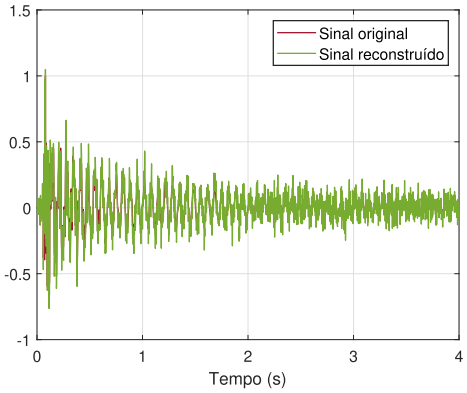
Fonte: Autor

Na terceira etapa, que corresponde à análise dos dados, a rede neural não supervisionada retorna os sinais reconstruídos. Na Figura 14(a) pode-se notar que a rede reconstrói bem o sinal utilizado no treinamento, sendo quase imperceptível a diferença entre os sinais de entrada e os de saída. Esse resultado se repete para os outros três modelos aqui apresentados.

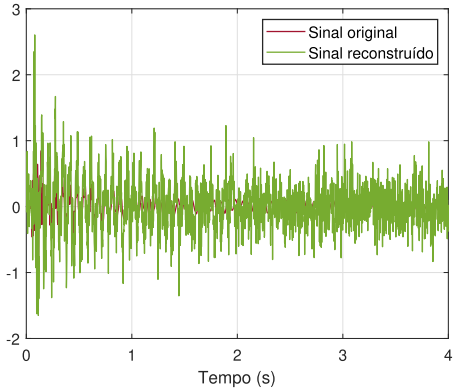
Durante a validação (Fig. 14(b)), nota-se que existe uma dispersão um pouco maior entre o sinal original e o reconstruído. Por fim, durante a fase de teste (sinais da estrutura com dano), percebe-se uma diferença considerável entre o original e o reconstruído. Esse padrão também foi observado ao longo dos testes realizados com todos os demais modelos.



(a) Fase de treinamento



(b) Fase de validação



(c) Fase de teste

Figura 14 – Modelo A: Exemplos de sinais originais e reconstruídos. Fonte: Autor.

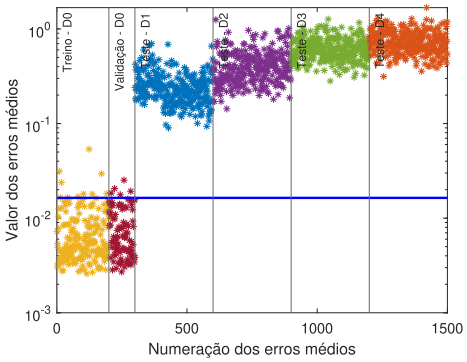
Uma vez os sinais reconstruídos pelo VAE, três métricas são utilizadas para se avaliar o erro entre os sinais originais e reconstruídos, além de se traçar um limiar acima de 95% dos resultados dos erros na fase de treinamento.

Esse processo é repetido dez vezes a fim de se avaliar a variabilidade de resultados em cada simulação. Os resultados de cada modelo, por acelerômetro, são mostrados nas Figuras 15, 16 e 17 (a abreviatura AC indica a palavra acelerômetro).

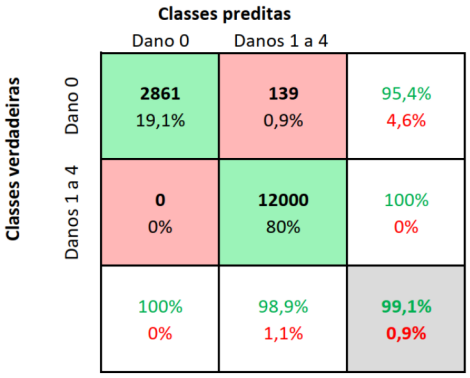
Para avaliar o desempenho do modelo de classificação criado, foram criadas matrizes de confusão. Esses elementos fornecem uma comparação detalhada entre a predição do modelo e o resultado real. Neste trabalho, as matrizes de confusão são geradas após 10 simulações (10-fold) em razão de se analisar a variabilidade dos resultados após cada repetição.

As Figuras 15(b,d,f,h), 16(b,d,f,h) e 17(b,d,f,h), apresentam as matrizes de confusão obtidas para cada acelerômetro. Nota-se que a acurácia dos três modelos estão próximas de 99%, indicando que eles conseguem "entender bem" os sinais de treinamento e, com isso, reconhecer valores similares e distinguir valores diferentes. Além disso, é possível visualizar nesses resultados que todos os modelos acertaram 100% os dados que continham dano. Já no caso dos dados que representam a estrutura sã, é possível visualizar que os modelos consideram aproximadamente 5% desses conjuntos com anomalia. Isso era de se esperar, haja vista que o limiar foi fixado justamente acima de 95% dos erros de reconstrução obtidos durante a fase de treino do algoritmo.

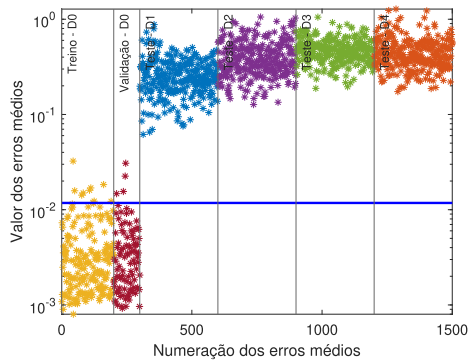
Entretanto, teoricamente, esperava-se que os gráficos mostrados nas Figuras 15(a,c,e,g), 16(a,c,e,g) e 17(a,c,e,g) produzissem uma espécie de "escada perfeita", na qual cada "degrau" representaria um cenário de dano diferente, indicando uma possível quantificação dos cenários de dano. Apesar de ter sido possível observar uma tendência similar em alguns casos (AC2 e AC4 - Modelo A; AC4 - Modelo B; todos AC Modelo C), para os demais isso não ocorreu.



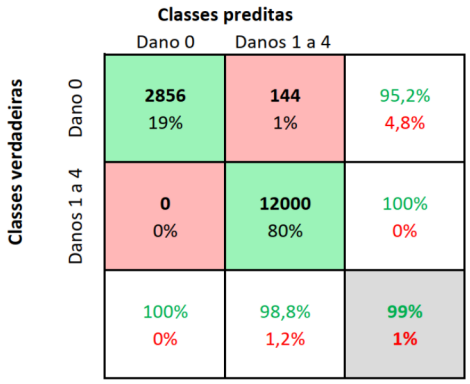
(a) AC1 - Erros de reconstrução



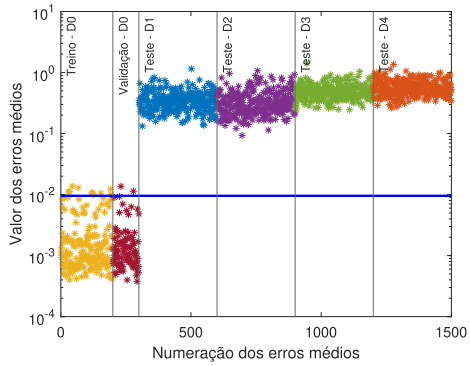
(b) AC1 - Matriz de confusão



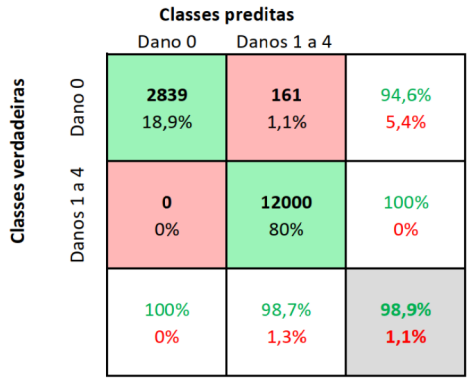
(c) AC2 - Erros de reconstrução



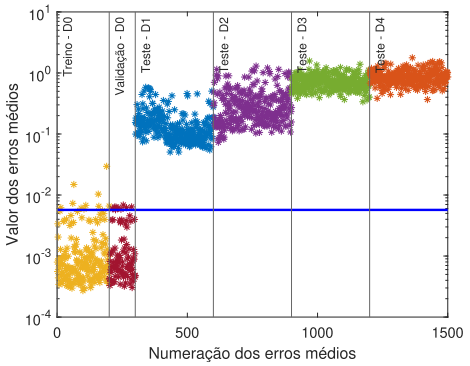
(d) AC2 - Matriz de confusão



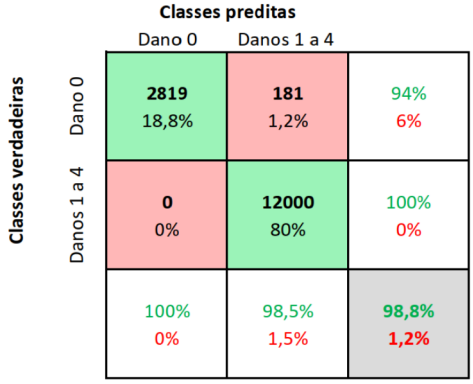
(e) AC3 - Erros de reconstrução



(f) AC3 - Matriz de confusão



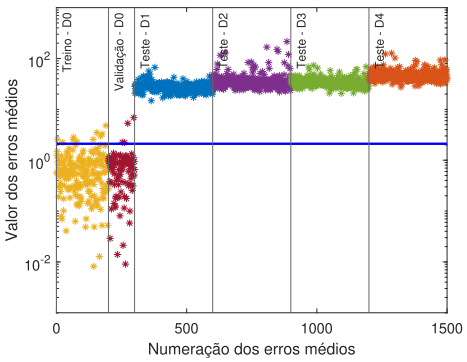
(g) AC4 - Erros de reconstrução



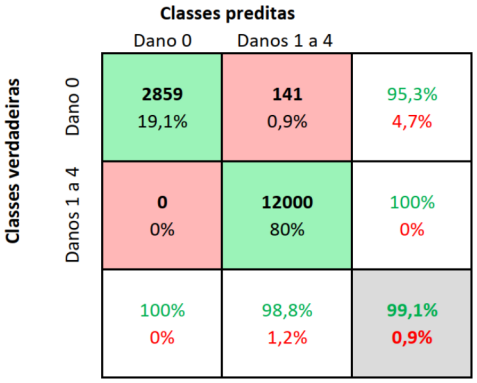
(h) AC4 - Matriz de confusão

Figura 15 – Resultados Modelo A

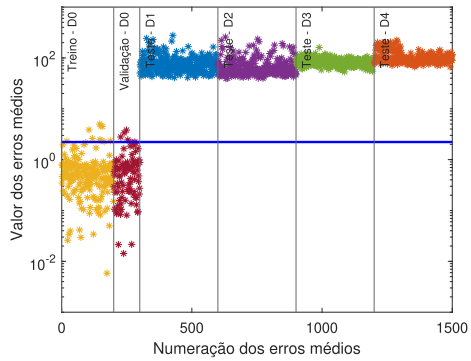
Fonte: Autor



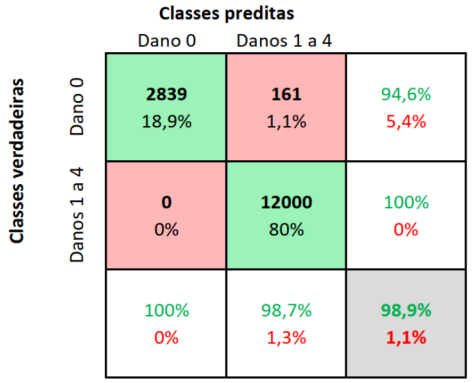
(a) AC1 - Erros de reconstrução



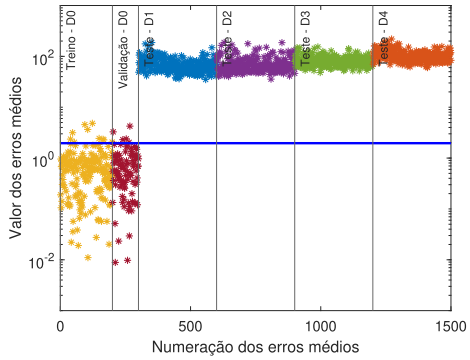
(b) AC1 - Matriz de confusão



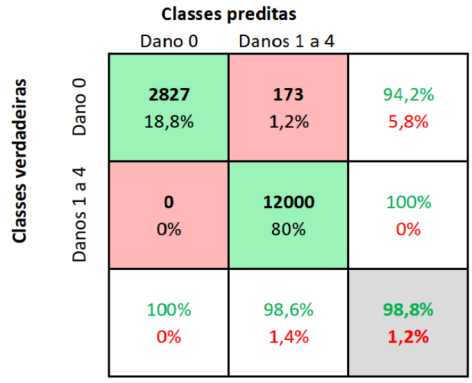
(c) AC2 - Erros de reconstrução



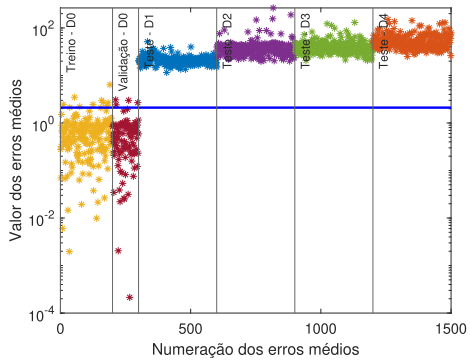
(d) AC2 - Matriz de confusão



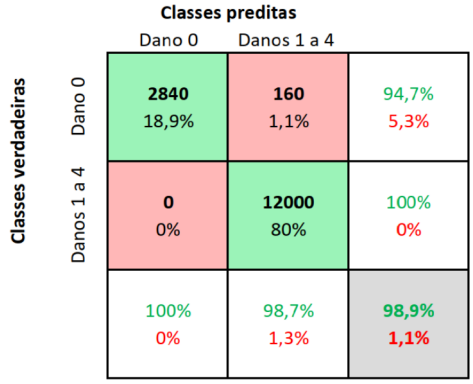
(e) AC3 - Erros de reconstrução



(f) AC3 - Matriz de confusão



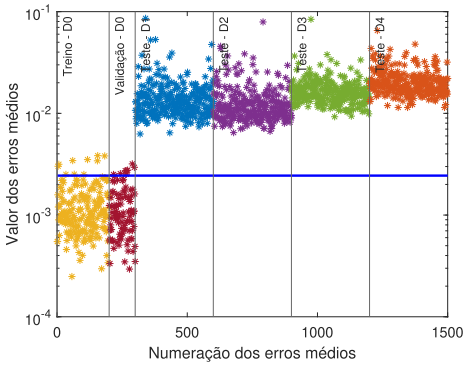
(g) AC4 - Erros de reconstrução



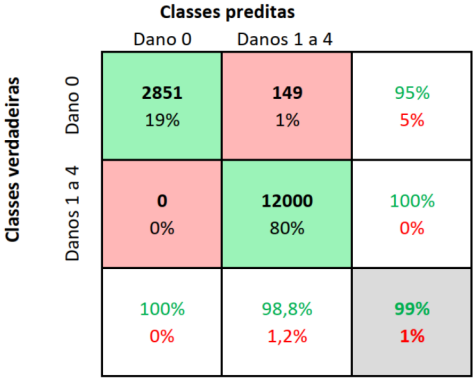
(h) AC4 - Matriz de confusão

Figura 16 – Resultados Modelo B

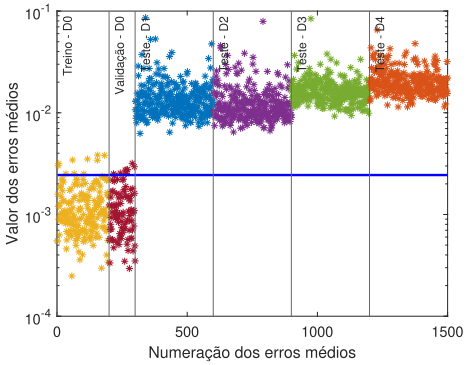
Fonte: Autor



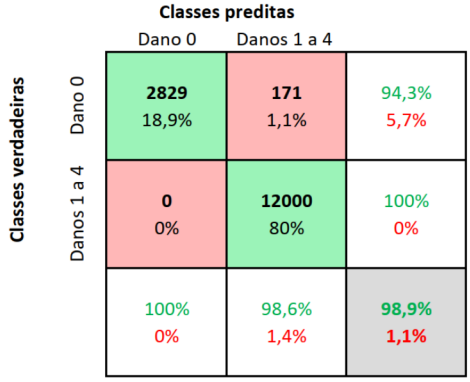
(a) AC1 - Erros de reconstrução



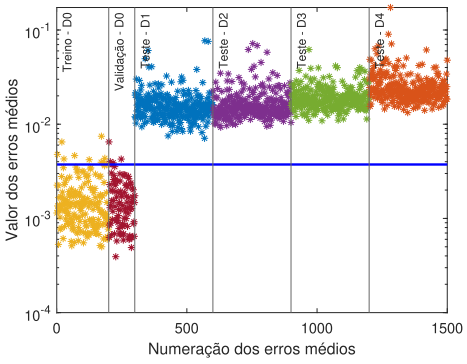
(b) AC1 - Matriz de confusão



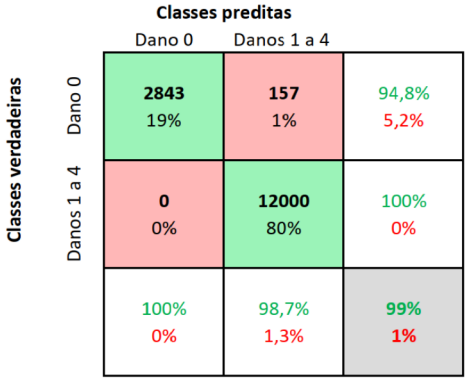
(c) AC2 - Erros de reconstrução



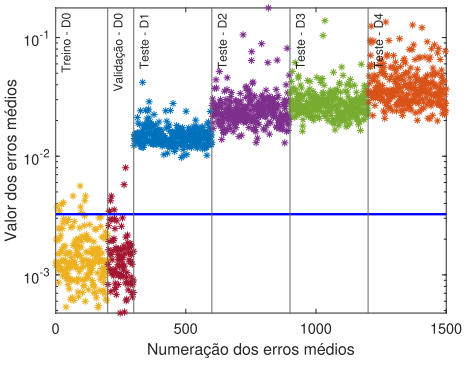
(d) AC2 - Matriz de confusão



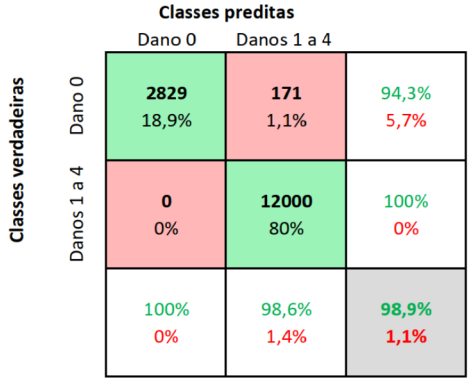
(e) AC3 - Erros de reconstrução



(f) AC3 - Matriz de confusão



(g) AC4 - Erros de reconstrução

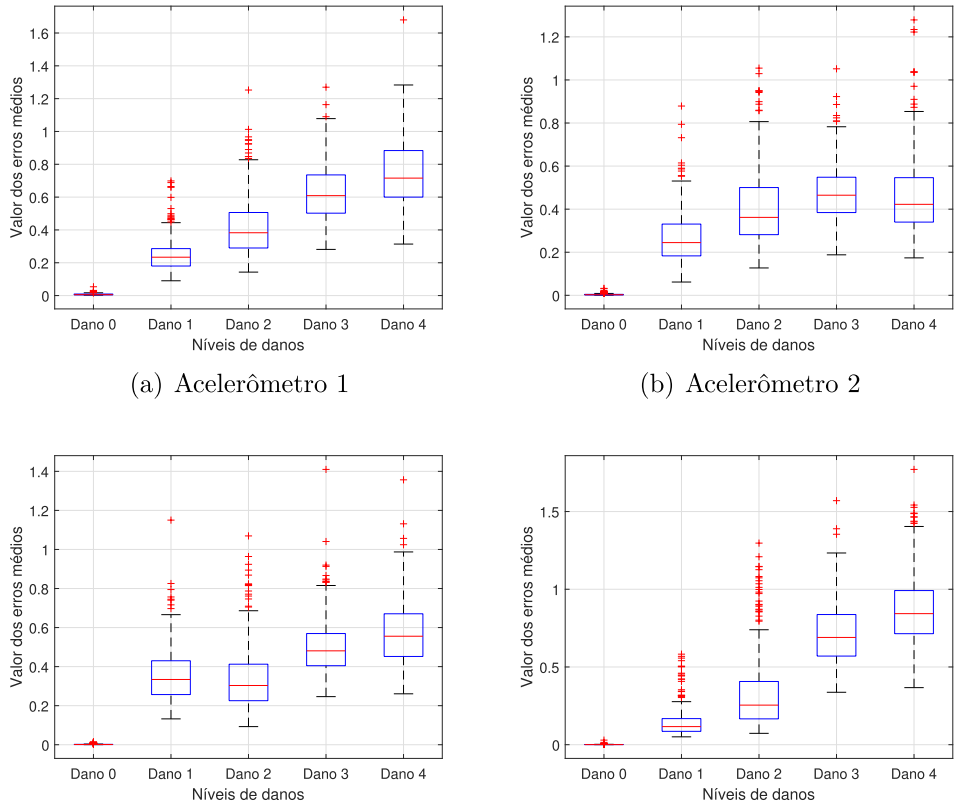


(h) AC4 - Matriz de confusão

Figura 17 – Resultados Modelo C

Fonte: Autor

Como forma de enriquecer essa análise, foram gerados diagramas de caixa (*box-plots*) específicos para cada modelo, considerando-se os dados de cada acelerômetro para os diferentes níveis de danos. Essas representações gráficas auxiliam na interpretação dos resultados obtidos a partir de um ponto de vista estatístico, pois permite avaliar tendências e possíveis valores discrepantes (*outliers*). Analisando-se os gráficos apresentados na Figuras 18, 19 e 20 pode-se perceber que a mediana relativa a cada cenário de dano aumenta, na maioria dos casos, à medida que se aumenta o nível de dano. No entanto, haja vista que existe uma superposição entre os "bigodes" de cada *box-plot*, não é possível afirmar que o modelo proposto consegue quantificar os diferentes cenários. Isto decorre do fato de que a dispersão entre os dados é alta, o que gera zonas de erros médios comuns a níveis de danos distintos.



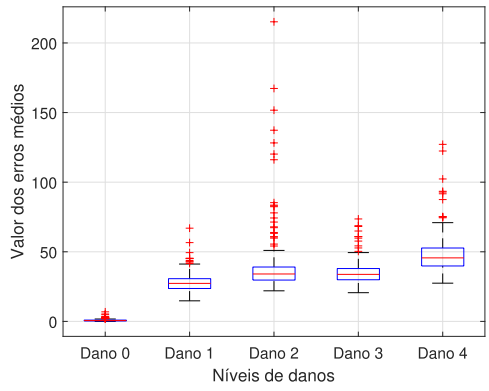
(a) Acelerômetro 1

(b) Acelerômetro 2

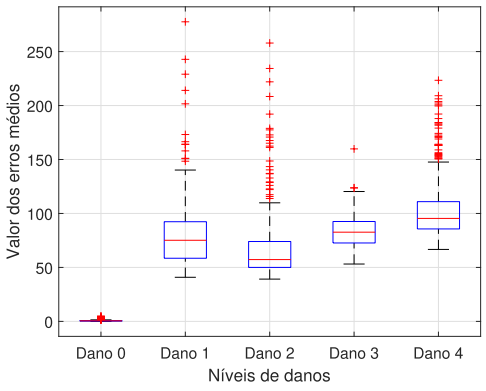
(c) Acelerômetro 3

(d) Acelerômetro 4

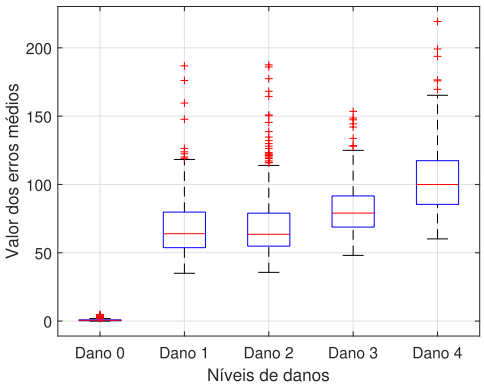
Figura 18 – Modelo A: Box-plot por acelerômetro, por nível de dano. Fonte: Autor.



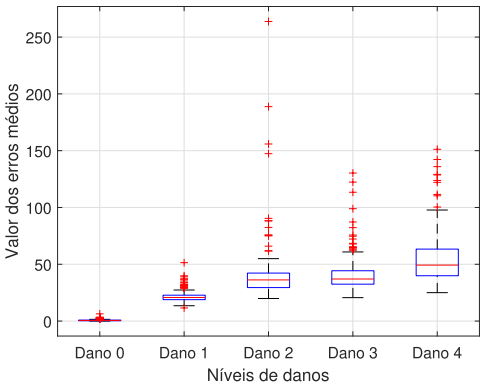
(a) Acelerômetro 1



(b) Acelerômetro 2

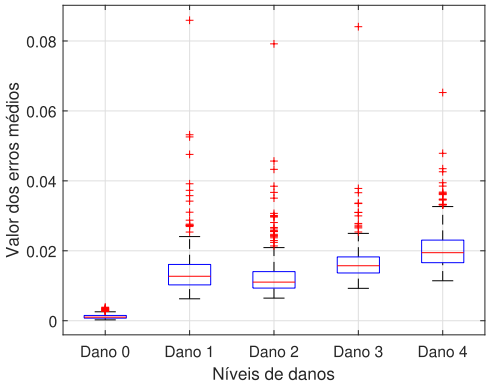


(c) Acelerômetro 3

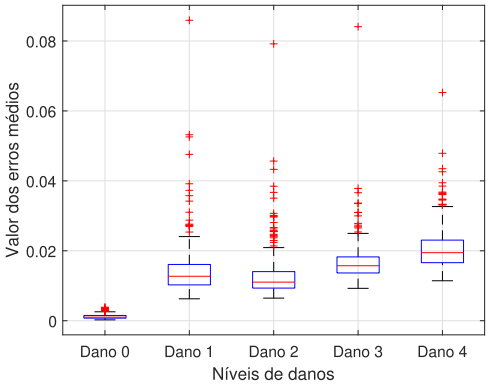


(d) Acelerômetro 4

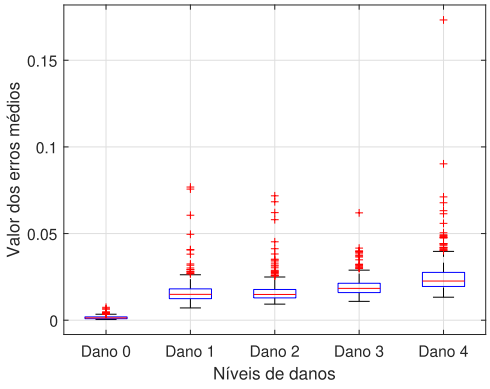
Figura 19 – Modelo B: Box-plot por acelerômetro, por nível de dano. Fonte: Autor.



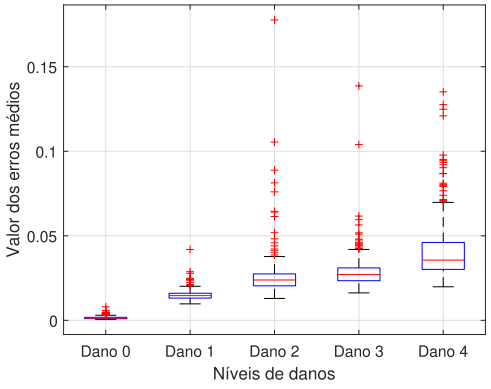
(a) Acelerômetro 1



(b) Acelerômetro 2



(c) Acelerômetro 3



(d) Acelerômetro 4

Figura 20 – Modelo C: Box-plot por acelerômetro, por nível de dano. Fonte: Autor.

Por fim, relativamente aos tempos médios de processamento para cada modelo, por acelerômetro, são: Modelo A - 6 horas; Modelo B - 8 horas; Modelo C - 12 horas.

4.2 PONTE Z24

A ponte Z24 (Figura 21), era uma ponte rodoviária localizada na Suíça, na região de Berna. A estrutura foi feita utilizando a técnica de concreto protendido, totalizando 58 metros de comprimento, distribuídos em três vãos contínuos - medindo 14, 30 e 14 metros respectivamente - conforme ilustrado na Figura 22. Essa obra de arte especial teve que ser demolida devido à construção de uma nova linha ferroviária. Antes de ser removida, a estrutura foi instrumentada e submetida a danos progressivos [50].

Figura 21 – Ponte Z24

Fonte: Maeck, Peeters e Roeck [51]

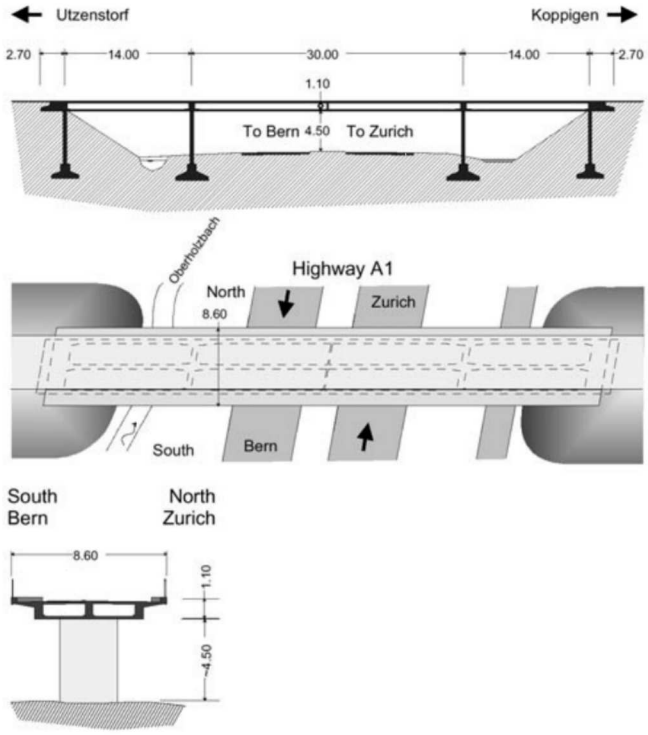


Figura 22 – Vistas esquemáticas da Z24 no sentido longitudinal, superior e transversal

Fonte: Maeck, Peeters e Roeck [51]

Dentre os testes realizados na ponte, em cada pilar foram instalados conjuntos de macacos hidráulicos que geravam recalques na estrutura. Foram quatro cenários de danos distintos e, para este trabalho, os sinais dinâmicos oriundos desses vibrações forçadas foram utilizados e são nomeados da seguinte forma:

- Dano 0: Estrutura intacta;
- Dano 1: Estrutura intacta após a instalação dos macacos hidráulicos em cada pilar;
- Dano 2: Recalque de 40 mm no pilar indicado na Figura 23;
- Dano 3: Recalque de 80 mm no pilar indicado na Figura 23.

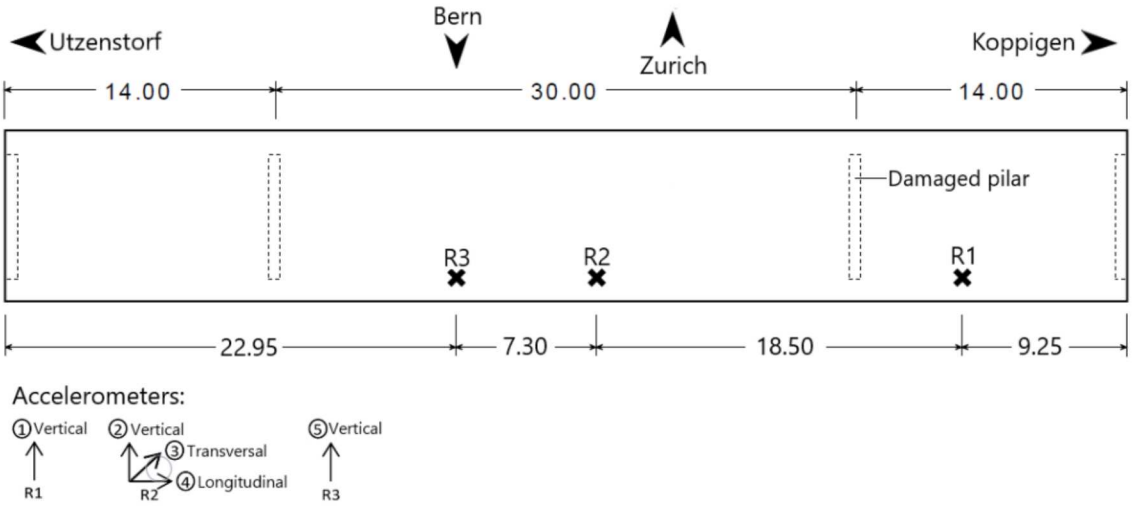
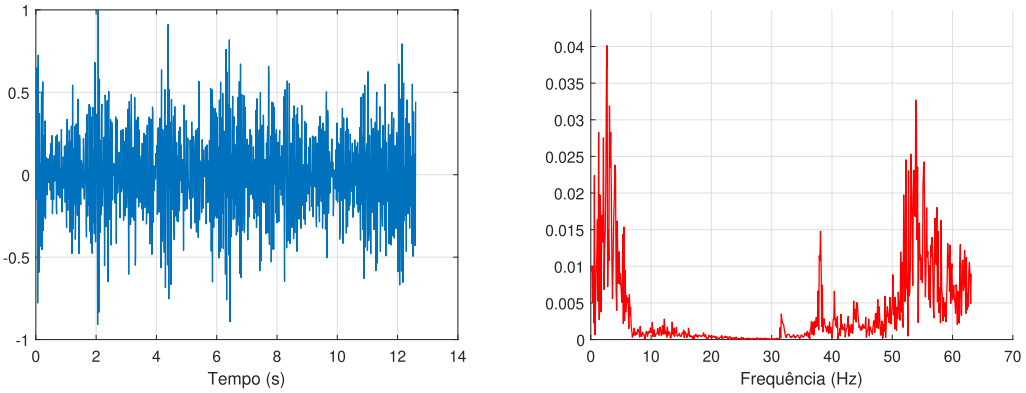


Figura 23 – Vista superior da ponte com o pilar danificado indicado

Fonte: Maeck, Peeters e Roeck [51]

Além disso, nessa instrumentação, dois *shakers* foram utilizados para gerar excitações verticais com a largura de banda variando entre 3 a 30 Hz. Para cada cenário de dano, foram realizados 9 ensaios com o auxílio de cinco acelerômetros dispostos em três pontos de medição (conforme mostra a Figura 23 como R1, R2 e R3). As respostas vibracionais possuem 65.835 pontos amostrados, coletados em aproximadamente 11 minutos a uma frequência de aquisição de 100 Hz. A Figura 24(a) mostra uma dessas respostas típicas no domínio do tempo e a Figura 24(b) mostra uma dessas respostas típicas no domínio da frequência (ambas normalizadas, conforme explicado no Cap. 3).

Com o intuito de tentar diferenciar os diferentes cenários de danos com o uso do VAE, os dados foram reorganizados, de modo que 65.530 pontos de cada um dos nove ensaios fossem alocados em 52 grupos contendo 1260 pontos. Dessa forma, para cada acelerômetro, há 468 (9 x 52) conjuntos de dados fornecidos como entrada para a RNA.



(a) No domínio do tempo (b) No domínio da frequência

Figura 24 – Resposta dinâmica típica da Z24

Fonte: Autor

De modo a analisar um maior número de redes possíveis, durante o período dessa pesquisa foram testados vários modelos presentes no Apêndice B (Seção 6.2). Dentro desse contexto, foram conduzidos 72 testes diferentes por acelerômetro. Entre os modelos há algumas variações: o domínio em que estão os dados de entrada (tempo, frequência e uma combinação de dados nesses dois domínios); intervalos diferentes para otimização dos neurônios das camadas de codificação e decodificação; alternância do número (uma ou duas) e tipo (FC, LSTM e Bi-LSTM) de camadas de ligação e tipos de função erros (MSE ou a união do MSE com a Distância de Mahalanobis). Como cada modelo demora cerca de 6 a 12 horas, esses testes foram feitos otimizando-se apenas o acelerômetro 3 e replicando-se os parâmetros para os demais.

4.2.1 Arquitetura das redes

Da mesma maneira que foi apresentada para a aplicação do pórtico, para cada tipo de ligação diferente entre os neurônios (FC, LSTM e Bi-LSTM) foi escolhida a melhor arquitetura (modelos D, E e F). Em todos os três estudos, a matriz de entrada no algoritmo é a mesma: para cada acelerômetro são 468 ensaios contendo 1260 pontos, por cenário. Desse modo, para os dados dos modelos D e

E, as entradas estão no domínio da frequência. Já para os dados do modelo F, o melhor domínio para detecção de dano foi o do tempo. A partir disso, após a otimização do modelo para cada acelerômetro, cada arquitetura passa pelo mesmo processo:

- **Fase de treinamento:** Inicialmente são treinadas utilizando os dados do cenário Dano 0. Assim, 300 ensaios foram separados para treinamento da rede e 168 para validação;
- **Fase de teste:** Posteriormente passam pela fase de teste, onde são testados 468 ensaios referentes a cada um dos outros modelos (cenários de danos 1, 2 e 3).

De modo similar ao pórtico, o que difere as redes testadas são os tipos de ligações de seus neurônios, aqui também foram coletadas as melhores para cada tipo. A arquitetura A é do tipo FC; a arquitetura B é do tipo LSTM); a arquitetura C é a do tipo Bi-LSTM.

Os parâmetros indicados no algoritmo, que se diferem daqueles utilizados no pórtico, são:

1. **Camadas do codificador:** Apenas para o modelo A foi necessário ter duas camadas no codificador, para os outros modelos foram utilizadas apenas uma. Para o modelo A as duas camadas camada são do tipo FC, já para o B foi utilizada a LSTM e para o C a Bi-LSTM;
2. **Camadas do decodificador:** Igual ao codificador;
3. **Neurônios do codificador:** O intervalo escolhido para otimização foi entre 300 e 400 neurônios para os modelos que entraram com os dados de entrada no domínio da frequência (Modelos D e E). Já para o modelo F, o intervalo escolhido foi o 500 a 600 neurônios. Esses intervalos são definidos como sendo aproximadamente a metade do número de características dos dados de entrada;
4. **Neurônios do decodificador:** Igual ao codificador;

5. **Número de variáveis na camada latente:** Após várias tentativas, os melhores resultados para o modelo A foi de 75 e para os modelos B e C foi de 100;

6. **Número de características:** Número retornado pelo programa de 631 para os modelos que os dados de entrada estão no domínio da frequência (D e E) e de 1260 para o modelo F, no qual os dados de entrada estão no domínio do tempo;

As demais etapas seguem de acordo com o explicado na seção anterior.

4.2.2 Resultados obtidos

As simulações relacionadas à ponte Z24 seguem o mesmo padrão explicado na aplicação do pórtico plano. A depender do modelo, os dados de entrada se referem aos sinais dinâmicos no domínio tempo ou no da frequência.

Assim, iniciada a fase de otimização da arquitetura da rede, o algoritmo retorna como primeiro resultado o gráfico mostrado na Figura 25. Para cada otimização realizada por acelerômetro, foi avaliado se o número de populações (inicialmente igual a 20) e o número de gerações (inicialmente igual a 10) estão adequados. Para todos as simulações, esses valores foram adequados.

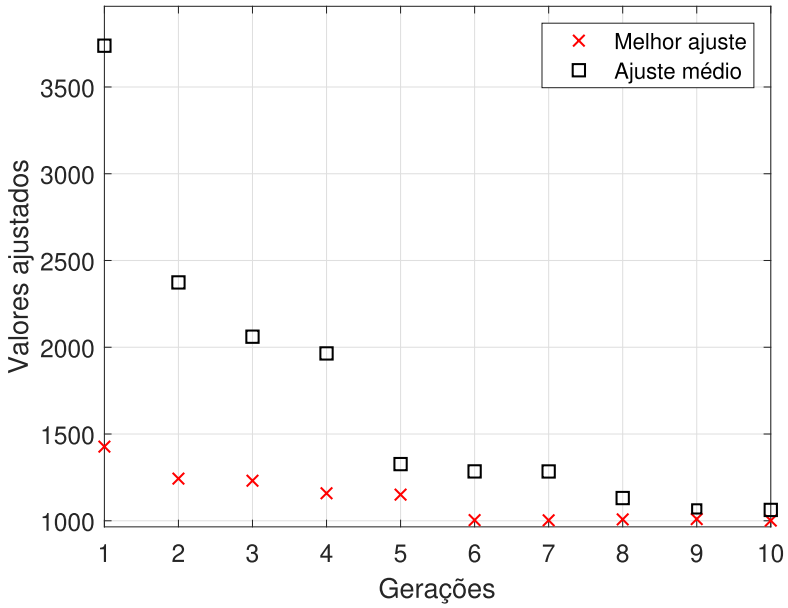


Figura 25 – Exemplo de um gráfico de gerações do modelo E - Acelerômetro 1

Fonte: Autor

Após essa etapa, o algoritmo retorna também os parâmetros otimizados da rede. As Figuras 26, 27 e 28 mostram os resultados obtidos por acelerômetro.

OTIMIZAÇÃO - Z24 - MODELO D				
Parâmetro	**AC1**	**AC2**	**AC3**	**AC4**
AutoencoderType	VAE	VAE	VAE	VAE
LayersEncoder	FC,FC	FC,FC	FC,FC	FC,FC
LayersDecoder	FC,FC	FC,FC	FC,FC	FC,FC
NeuronsEncoder	[387 336]	[360 371]	[370 350]	[369 331]
NeuronsDecoder	[380 305]	[324 320]	[370 306]	[382 353]
LatentDim	75	75	75	75
NumberEpoch	99	73	91	97
NumberFeature	631	631	631	631
LearningRate	0,0008491	0,0008916	0,0008196	0,0008745
MiniBatchSize	36	4	2	78
ExecutionEnvironment	AUTO	AUTO	AUTO	AUTO
WeightingKL	52	65	63	57
OutputTransferFunction	NONE	NONE	NONE	NONE
Domínio	FREQUÊNCIA	FREQUÊNCIA	FREQUÊNCIA	FREQUÊNCIA
Classificação	MSE	MSE	MSE	MSE

Figura 26 – Parâmetros otimizados, por acelerômetro, Modelo D

OTIMIZAÇÃO - Z24 - MODELO E				
Parâmetro	**AC1**	**AC2**	**AC3**	**AC4**
AutoencoderType	VAE	VAE	VAE	VAE
LayersEncoder	LSTM	LSTM	LSTM	LSTM
LayersDecoder	LSTM	LSTM	LSTM	LSTM
NeuronsEncoder	394	381	312	381
NeuronsDecoder	314	367	312	367
LatentDim	100	100	100	100
NumberEpoch	100	100	86	100
NumberFeature	631	631	631	631
LearningRate	0,0009499	0,0009499	0,0008443	0,0009499
MiniBatchSize	64	40	2	40
ExecutionEnvironment	AUTO	AUTO	AUTO	AUTO
WeightingKL	79	76	36	76
OutputTransferFunction	NONE	NONE	NONE	NONE
Domínio	FREQUENCIA	FREQUÊNCIA	FREQUENCIA	FREQUÊNCIA
Classificação	MSE	MSE	MSE	MSE

Figura 27 – Parâmetros otimizados, por acelerômetro, Modelo E

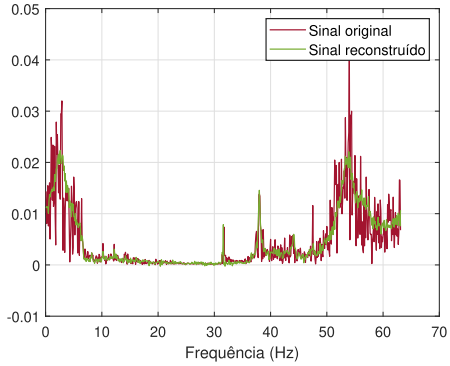
OTIMIZAÇÃO - Z24 - MODELO F				
Parâmetro	**AC1**	**AC2**	**AC3**	**AC4**
AutoencoderType	VAE	VAE	VAE	VAE
LayersEncoder	Bi-LSTM	Bi-LSTM	Bi-LSTM	Bi-LSTM
LayersDecoder	Bi-LSTM	Bi-LSTM	Bi-LSTM	Bi-LSTM
NeuronsEncoder	527	516	552	576
NeuronsDecoder	545	559	512	566
LatentDim	100	100	100	100
NumberEpoch	86	57	73	82
NumberFeature	1260	1260	1260	1260
LearningRate	0,0005324	0,0008808	0,000518	0,0007659
MiniBatchSize	50	24	100	64
ExecutionEnvironment	AUTO	AUTO	AUTO	AUTO
WeightingKL	0	90	3	9
OutputTransferFunction	NONE	NONE	NONE	NONE
Domínio	TEMPO	TEMPO	TEMPO	TEMPO
Classificação	MSE+MAHALA	MSE+MAHALA	MSE+MAHALA	MSE+MAHALA

Figura 28 – Parâmetros otimizados, por acelerômetro, Modelo F

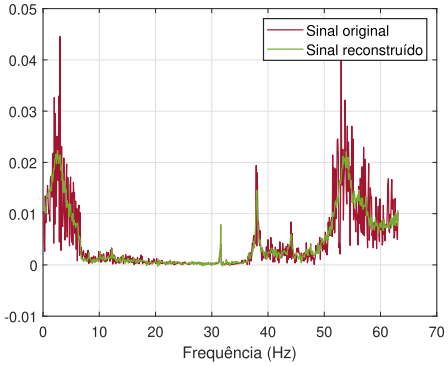
Fonte: Autor

Na terceira etapa, que corresponde à análise dos dados, a rede neural não supervisionada retorna os sinais reconstruídos. Da mesma forma como foi observado na aplicação anterior, todos os modelos reconstroem relativamente bem as tendências dos sinais utilizados no treinamento, validação e teste, ainda que se notem diferenças entre os sinais de entrada e os de saída (Figuras 29 e 30).

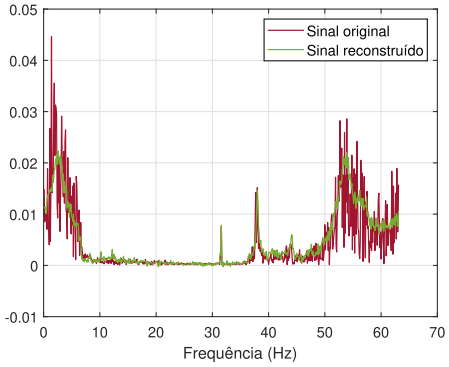
Entretanto, conforme será discutido adiante, o Modelo F (Figura 31), apesar de treinar bem com os dados, apresenta um um sobreajuste (*overfitting*) na validação. Isso significa que o modelo "decora" o padrão dos dados, ao invés de aprender, efetivamente.



(a) Fase de treinamento

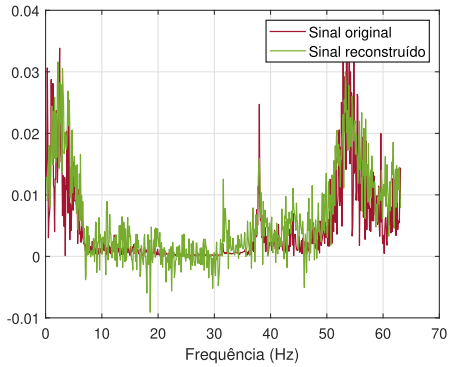


(b) Fase de validação

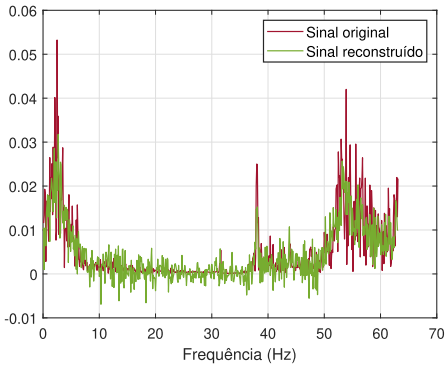


(c) Fase de teste

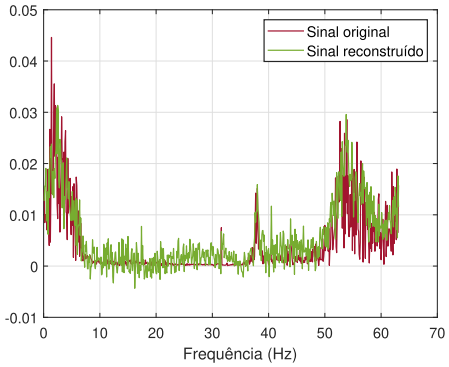
Figura 29 – Modelo D: Exemplos de sinais originais e reconstruídos. Fonte: Autor.



(a) Fase de treinamento

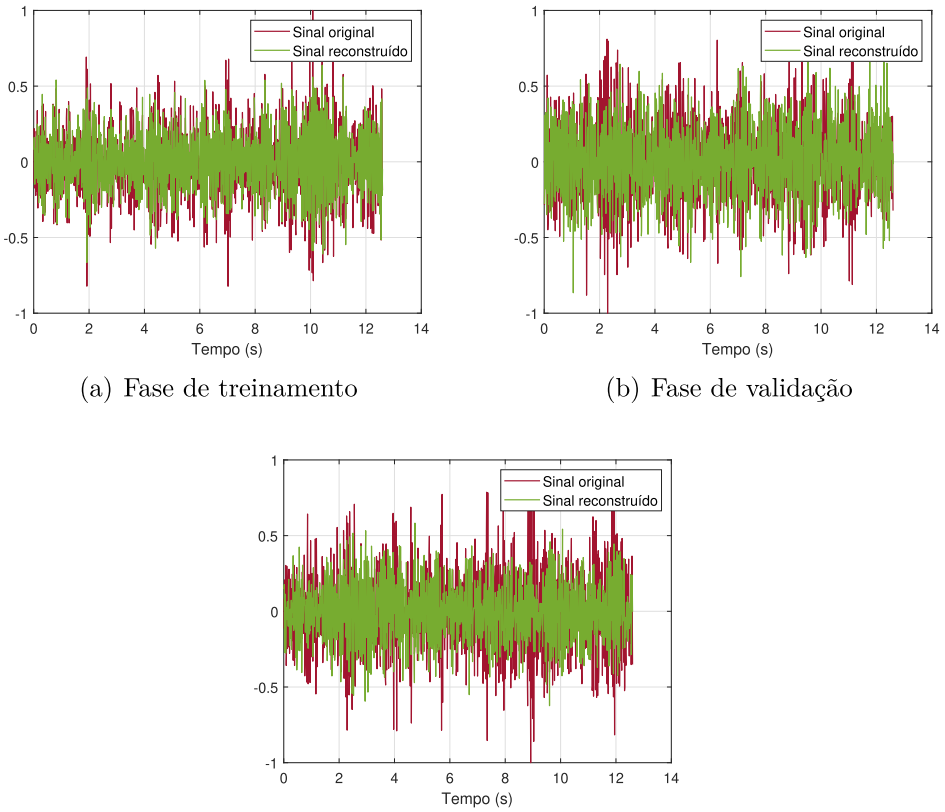


(b) Fase de validação



(c) Fase de teste

Figura 30 – Modelo E: Exemplos de sinais originais e reconstruídos. Fonte: Autor.



(a) Fase de treinamento

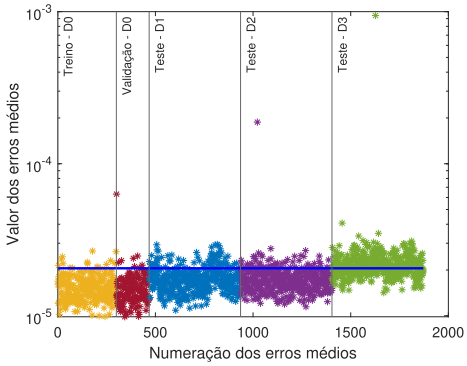
(b) Fase de validação

(c) Fase de teste

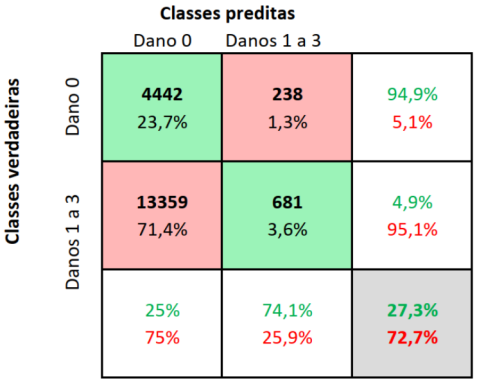
Figura 31 – Modelo F: Exemplos de sinais Original e Reconstruído

Fonte: Autor

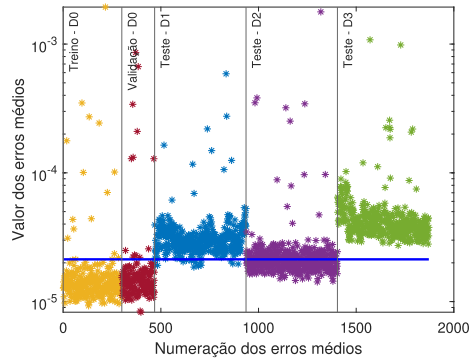
Uma vez os sinais reconstruídos pelo VAE, as mesmas métricas são utilizadas para se avaliar o erro entre os sinais originais e reconstruídos, além de se traçar um limiar acima de 95% dos resultados dos erros na fase de treinamento. Os resultados de cada modelo, por acelerômetro, são ilustrados nas Figuras 32, 33 e 34. As matrizes de confusão são os resultados após as 10 simulações de validação (10-fold) em razão de se analisar a variabilidade dos resultados após cada repetição.



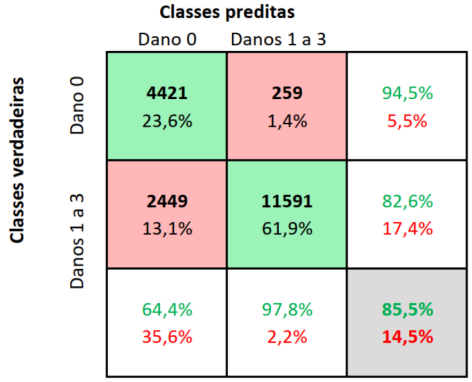
(a) AC1 - Erros de reconstrução



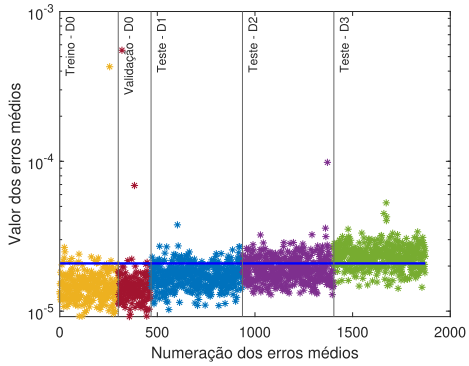
(b) AC 1 - Matriz de confusão



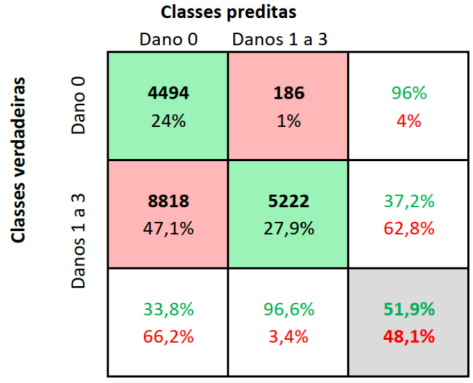
(c) AC2 - Erros de reconstrução



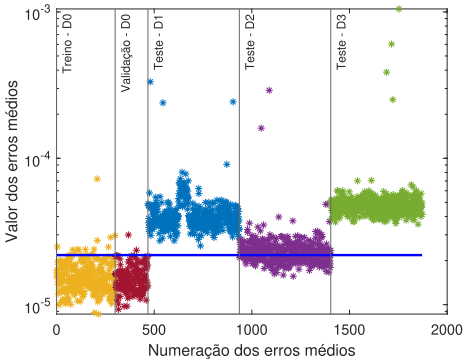
(d) AC 2 - Matriz de confusão



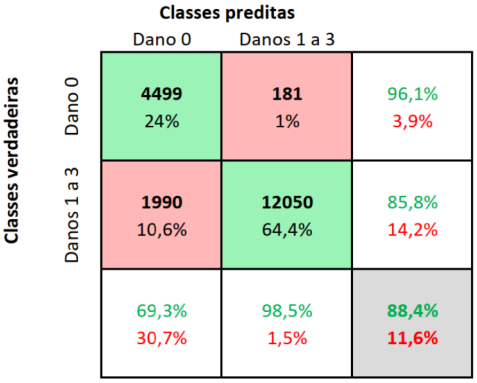
(e) AC3 - Erros de reconstrução



(f) AC 3 - Matriz de confusão



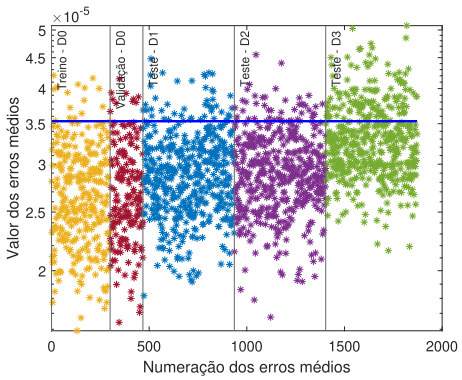
(g) AC4 - Erros de reconstrução



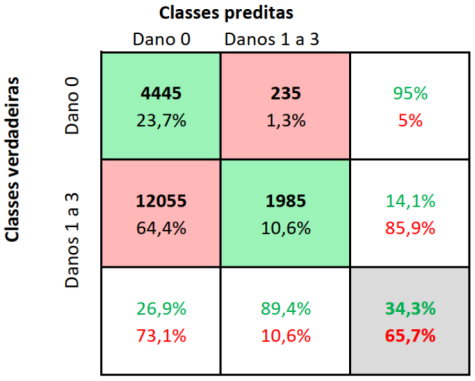
(h) AC 4 - Matriz de confusão

Figura 32 – Resultados Modelo D

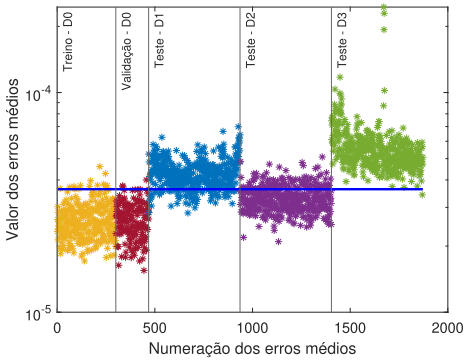
Fonte: Autor



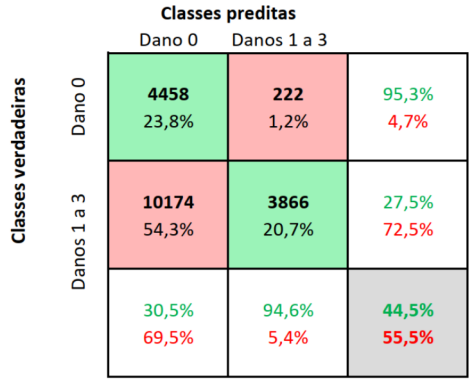
(a) AC1 - Erros de reconstrução



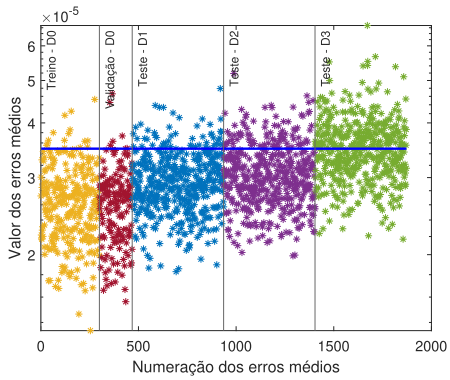
(b) AC 1 - Matriz de confusão



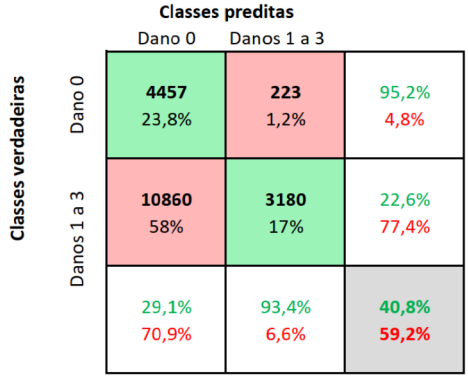
(c) AC2 - Erros de reconstrução



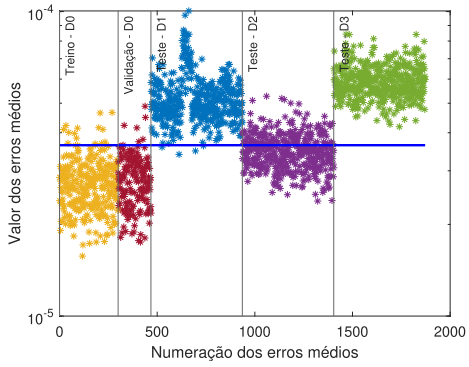
(d) AC 2 - Matriz de confusão



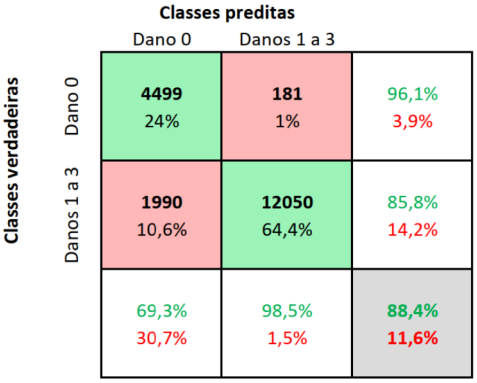
(e) AC3 - Erros de reconstrução



(f) AC 3 - Matriz de confusão



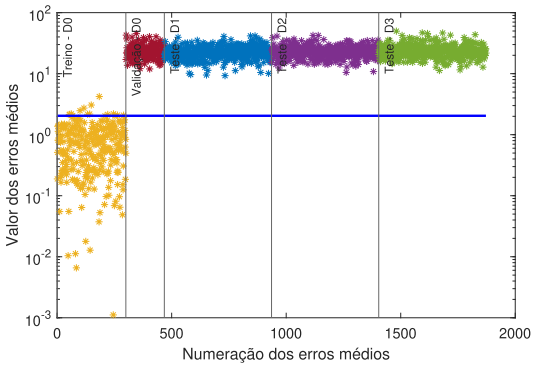
(g) AC4 - Erros de reconstrução



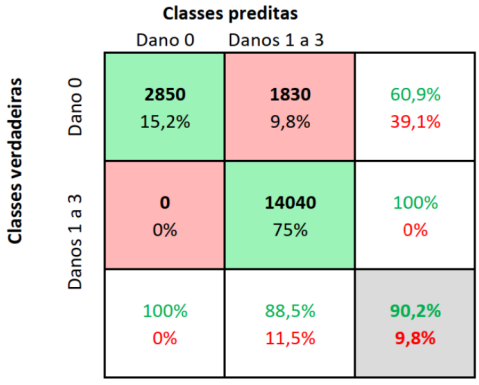
(h) AC 4 - Matriz de confusão

Figura 33 – Resultados Modelo E

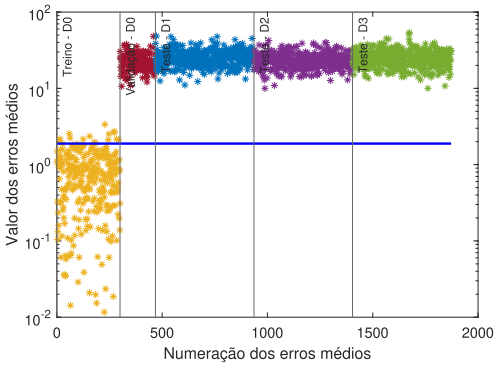
Fonte: Autor



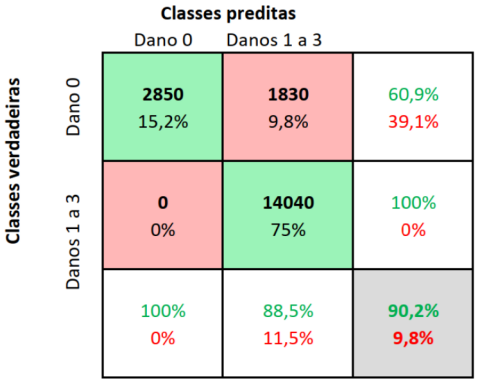
(a) AC1 - Erros de reconstrução



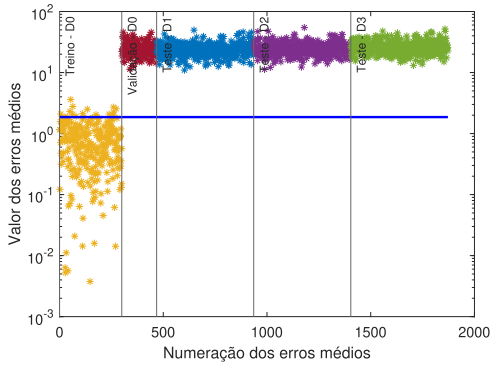
(b) AC 1 - Matriz de confusão



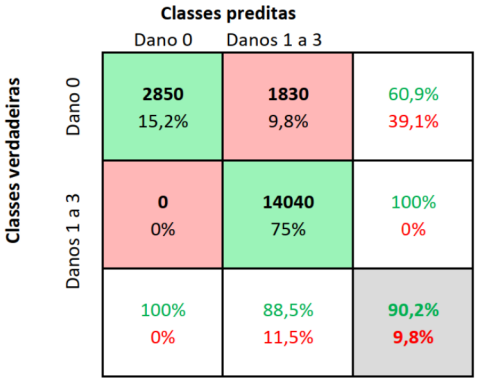
(c) AC2 - Erros de reconstrução



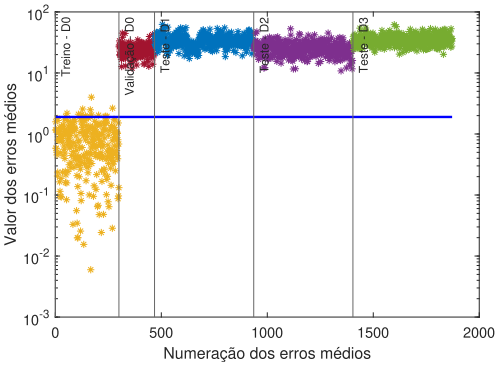
(d) AC 2 - Matriz de confusão



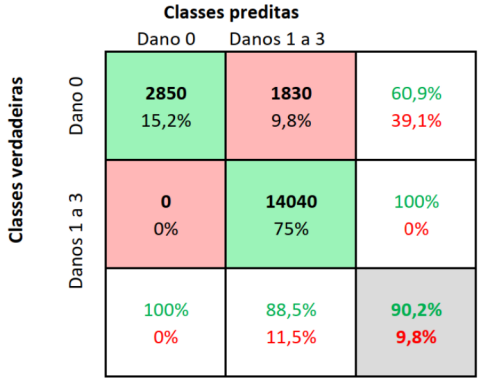
(e) AC3 - Erros de reconstrução



(f) AC 3 - Matriz de confusão



(g) AC4 - Erros de reconstrução



(h) AC 4 - Matriz de confusão

Figura 34 – Resultados Modelo F

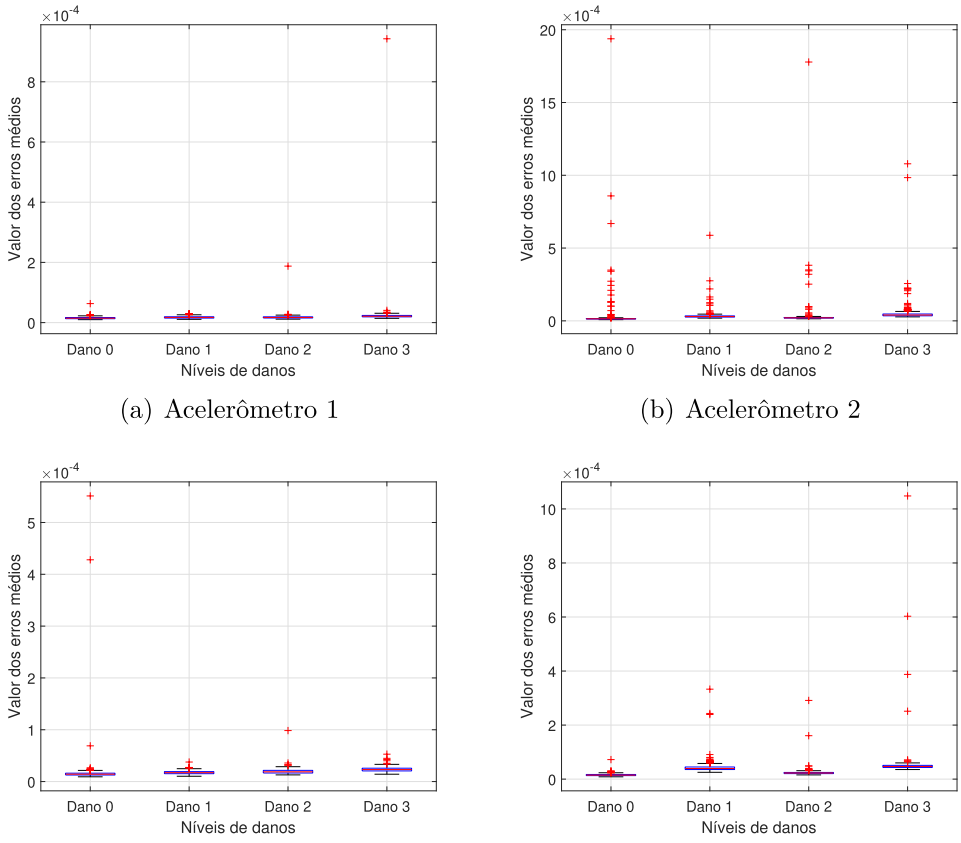
Fonte: Autor

Conforme mostrado nas Figuras 32 e 33, observa-se que, para os modelos D e E, os resultados não atenderam às expectativas. As taxas de acurácia apresentaram variações significativas entre os acelerômetros, não permitindo conclusões acerca da eficácia dos modelos no que tange à detecção de dano. A metodologia proposta não conseguiu distinguir adequadamente as anomalias, embora, de maneira superficial, tenha indicado uma propensão à combinação dos erros provenientes dos acelerômetros 2 e 4. Para o modelo D, os *box-plots* (Figura 35) tampouco se mostraram conclusivos, visto que apesar de os dados possuírem uma pequena dispersão, muitos deles são indicados como *outliers*, além de existir superposição dos "bigodes" de cada *box-plot*.

Para o modelo E, é possível visualizar na Figura 36 que os dados dos acelerômetros 1 e 3 apresentam uma tendência crescente dos valores da mediana, à medida que se aumenta o nível de dano. Entretanto, como a dispersão entre os dados é alta, há zonas de erros médios comuns à danos distintos. Para os acelerômetros 2 e 4, a dispersão entre os dados de cada dano é menor, mas a mediana não aumenta à medida que o nível de dano é aumentado.

De forma análoga ao Pórtico, para os modelos D e E, no caso dos dados que representam a estrutura sã, é possível visualizar que os modelos consideram aproximadamente 5% desses conjuntos com anomalia. Isso era de se esperar, haja vista que o limiar foi fixado justamente acima de 95% dos erros de reconstrução obtidos durante a fase de treino do algoritmo.

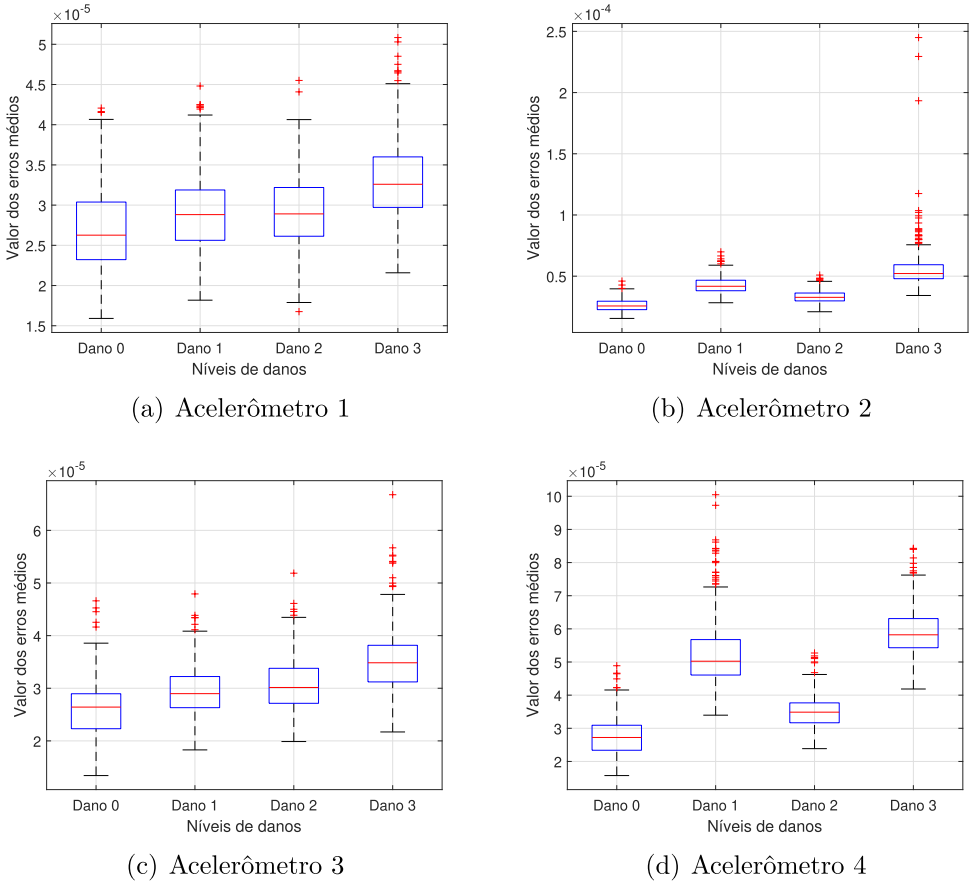
No caso do modelo F é perceptível visualmente o sobreajuste nos dados de treinamento, uma vez que a metodologia não consegue distinguir entre novas entradas semelhantes ou distintas (note que os próprios dados de validação são classificados como "danificados"). Os *box-plots* apresentados na Figura 37 mostram que as medianas dos erros médios são similares para os casos em que a estrutura apresenta dano. No estado íntegro, o valor da mediana é baixo, mas a dispersão é alta.



(a) Acelerômetro 1 (b) Acelerômetro 2

(c) Acelerômetro 3 (d) Acelerômetro 4

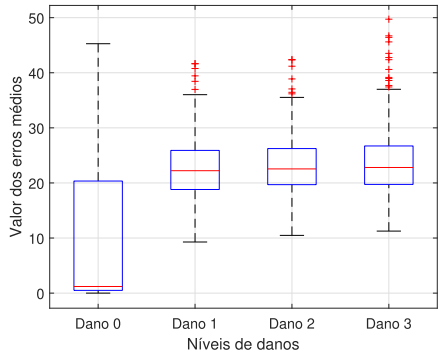
Figura 35 – Modelo D: Box-plot por acelerômetro, por nível de dano. Fonte: Autor.



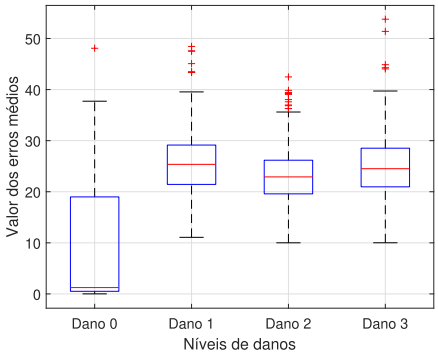
(a) Acelerômetro 1 (b) Acelerômetro 2

(c) Acelerômetro 3 (d) Acelerômetro 4

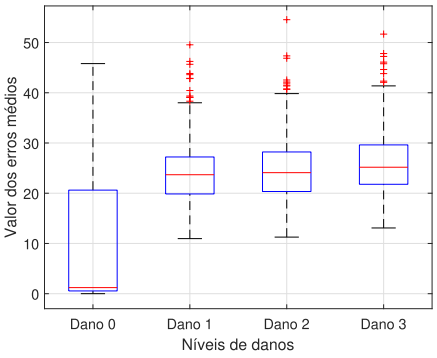
Figura 36 – Modelo E: Box-plot por acelerômetro, por nível de dano. Fonte: Autor.



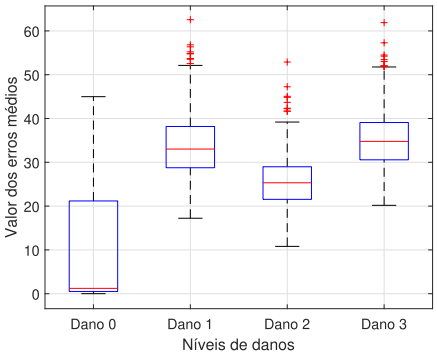
(a) Acelerômetro 1



(b) Acelerômetro 2



(c) Acelerômetro 3



(d) Acelerômetro 4

Figura 37 – Modelo F: Box-plot por acelerômetro, por nível de dano

Fonte: Autor

Por fim, os tempos médios de processamento por acelerômetro, para cada modelo, foram: 10 horas para o Modelo D, 13 horas para o Modelo E e 16 horas para Modelo o F.

5 CONCLUSÕES E TRABALHOS FUTUROS

O presente trabalho teve por objetivo avaliar a eficiência do VAE em problemas de SHM no que tange ao reconhecimento de danos em estruturas civis. Dessa forma, forram avaliados parâmetros que definem o comportamento do VAE e sua capacidade de reproduzir sinais dinâmicos em uma abordagem não supervisionada. Para isso, foi criado um código com o auxílio e adaptações realizadas no algoritmo proposto pelos autores Terbuch [17] e Terbuch *et al.* [41].

A metodologia proposta foi validada em duas aplicações experimentais: em um pórtico ensaiado no Laboratório de Imagens e Sinais da UFJF, e no clássico caso da Ponte Z24, na Suíça.

Diversos modelos foram avaliados em relação ao pórtico, sendo selecionados os três mais promissores. As acurácias médias ficaram em torno de 99% para todos eles. Dessa forma, os resultados obtidos foram considerados satisfatórios, haja vista que o algoritmo foi capaz de identificar e retornar uma boa tendência de separação dos danos em relação ao limiar pré-fixado. No entanto, ainda não é possível afirmar que o modelo consegue quantificar os diferentes níveis de danos.

Já para a aplicação da ponte Z24, dois dos três melhores modelos (D e E), o algoritmo não separou bem os diferentes cenários danos, ainda que tenha sido possível observar uma leve tendência de classificação correta, principalmente para os acelerômetros 2 e 4. Entretanto, para o modelo F, o algoritmo treina bem, mas não reconhece novas entradas, sinais de um sobreajuste de dados.

Como considerações finais, destaca-se:

- A rede neural com o VAE consegue treinar bem as respostas dinâmicas dadas como entrada;
- O resultado reconstruído pode ser influenciado pelo domínio dos dados de entrada, dependendo da estrutura considerada;
- A rede neural consegue reconstruir bem, mesmo com ruído, os tipos de danos distintos ao se entrar com dados brutos em testes feito em um ambiente controlado;

- Em aplicações com um ambiente mais controlado o algoritmo consegue não só detectar a presença do dano, como também apresenta uma tendência de quantificação;
- Em aplicações com dados obtidos de estruturas de grande porte, o algoritmo conseguiu treinar bem os dados, mas não reconheceu bem novas entradas;
- Apesar de os resultados em estruturas de grande porte não serem satisfatórios, o algoritmo apresenta uma certa tendência de concentração dos erros de cenários distintos.

Assim, como foi visto no presente trabalho, o VAE se comportou bem ao serem inseridas respostas dinâmicas. Além disso, foi possível identificar danos em estruturas que foram coletados dados em um ambiente controlado. No entanto, para estruturais reais o modelo não se comportou de uma forma adequada, assim objetiva-se em trabalhos futuros:

- Testar novos limiares para realizar a classificação dos dados;
- Deve-se testar novos métodos que encontrem de modo otimizado e automático os melhores intervalos a serem inseridos para os parâmetros durante a fase de otimização;
- Deve-se averiguar uma possibilidade de se rodar o algoritmo utilizando computação paralela e GPU;
- Utilizar as características reduzidas da camada latente como indicadores de dano;
- Realizar novas aplicações do algoritmo em outras estruturas reais a fim de observar seu comportamento.

REFERÊNCIAS

1 FINOTTI, R. P. **Inteligência artificial aplicada ao monitoramento de estruturas: detecção de alterações mecânico-estruturais baseada no uso de redes neurais autocodificadoras esparsas para a caracterização de respostas dinâmicas**. 169 p. Tese (Doutorado) — Universidade Federal de Juiz de Fora, Juiz de Fora, Brasil, 2022.

2 CARDOSO, A. C. Rharã de A.; BARBOSA, F. Automated real-time damage detection strategy using raw dynamic measurements. **Engineering Structures**, Elsevier, v. 196, p. 1–16, out. 2019. ISSN 1557-9662.

3 LOPES, P. C. F.; DIAS, A. (des)territórios da mineração: planejamento territorial a partir do rompimento em mariana, mg. **Cadernos Metrópole**, Pontifícia Universidade Católica de São Paulo, v. 19, n. 38, p. 282–283, jan. 2017. ISSN 1517-2422.

4 BATTISTA, R. C.; PFEIL, M. S. Reduction of vortex-induced oscillations of rio–niterói bridge by dynamic control devices. **Journal of Wind Engineering and Industrial Aerodynamics**, Elsevier, v. 84, n. 3, p. 273–288, 2000.

5 NI, Y.; XIA, H.; WONG, K.; KO, J. In-service condition assessment of bridge deck using long-term monitoring data of strain response. **Journal of Bridge Engineering**, American Society of Civil Engineers, v. 17, n. 6, p. 876–885, 2012.

6 CACHOT, E.; VAYSSADE, T.; VIRLOGEUX, M.; LANCON, H.; HAJAR, Z.; SERVANT, C. The millau viaduct: ten years of structural monitoring. **Structural Engineering International**, Taylor & Francis, v. 25, n. 4, p. 375–380, 2015.

7 RIERA, J. D. Sobre a definição do vento para projeto estrutural na abnt nbr-6123 (1989) e outras normas sul americanas. **Revista Sul-Americana de Engenharia Estrutural**, v. 13, n. 3, 2016.

8 ABNT. Nbr 6123: Forças devido ao vento em edificações. **Associação Brasileira de Normas Técnicas**, SI, 2023.

9 ABNT. Nbr 15421: Projeto de estruturas resistentes a sismo. **Associação Brasileira de Normas Técnicas**, SI, 2023.

10 ABNT. Nbr 8681: Ações e segurança na estrutura - procedimento. **Associação Brasileira de Normas Técnicas**, SI, 2003.

11 UBC. International conference of building officials. **Uniform Building Code**, v. 2, 1997.

12 EUROCODE. Eurocódigo 8: Projecto de estruturas para resistência aos sismos - parte 1: regras gerais, ações sísmicas e regras para edificios. **Comitê Europeu de Normalização**, v. 8, 2004.

13 CREMONA, C.; SANTOS, J. P. Structural health monitoring as a big-data problem. **Structural Engineering International**, Structural Engineering International, v. 1, n. 9, p. 243–254, jan. 2018. ISSN 1683-0350.

14 NUNES, L. A.; AMARAL, R. P. F.; BARBOSA, F. d. S.; CURY, A. A. A hybrid learning strategy for structural damage detection. **Structural Health Monitoring**, SAGE Publications Sage UK: London, England, v. 20, n. 4, p. 2143–2160, 2021.

15 SHANG, Z.; SUN, L.; XIA, Y.; ZHANG, W. Vibration-based damage detection for bridges by deep convolutional denoising autoencoder. **Structural Health Monitoring**, SAGE Publications Sage UK: London, England, v. 20, n. 4, p. 1880–1903, 2021.

16 BRAGA, A. de P.; LUDERMIR, T. B.; FERREIRA, A. C. P. de L.; FERREIRA, A. C. P. de L. **Redes neurais artificiais: teoria e aplicaçõe**. [S.l.]: Rio de Janeiro: LTC, 2000.

17 TERBUCH, A. Lstm hyperparameter optimization: Impact of the selection of hyperparameters on machine learning performance when applied to time series in physical systems. 2021.

18 AMARAL, R. P. F.; BARBOSA, F. de S.; CURY, A. A.; FONSECA, L. G. da; BONIFÁCIO, A. L. Aplicação de métodos computacionais a dados vibracionais para detecção de alterações estruturais. **XII SIMPÓSIO DE MECÂNICA COMPUTACIONAL**, 2016.

19 FINOTTI, R. P.; GENTILE2B, C.; BARBOSA, F.; CURY, A. Structural novelty detection based on sparse autoencoders and control charts. **Structural Engineering and Mechanics**, v. 81, n. 5, p. 647–664, 2022.

20 AN, J.; CHO, S. Variational autoencoder based anomaly detection using reconstruction probability. **Special Lecture on IE**, Special Lecture on IE, v. 2015-2, p. 1–18, 2015.

21 CAWLEY, P.; ADAMS, R. D. The location of defects in structures from measurements of natural frequencies. **The Journal of Strain Analysis for Engineering Design**, SAGE Publications Sage UK: London, England, v. 14, n. 2, p. 49–57, 1979.

22 FOX, C. The location of defects in structures-a comparison of the use of natural frequency and mode shape data. In: **10th International modal analysis conference**. [S.l.: s.n.], 1992. v. 1, p. 522–528.

23 SHA, G.; RADZIEŃSKI, M.; CAO, M.; OSTACHOWICZ, W. A novel method for single and multiple damage detection in beams using relative natural frequency changes. **Mechanical Systems and Signal Processing**, Elsevier, v. 132, p. 335–352, 2019.

24 ALLEMANG, R. J. A correlation coefficient for modal vector analysis. In: **Proc. of the 1st IMAC**. [S.l.: s.n.], 1982. p. 110–116.

25 LIEVEN, N.; EWINS, D. Spatial correlation of mode shapes, the coordinate modal assurance criterion (comac). **Proceedings of the sixth international modal analysis conference**, Proceedings of the sixth international modal analysis conference, v. 1, p. 110–116, 1982.

26 CURY, A. **Techniques d'anormalité appliquées à la surveillance de santé structurale**. Tese (Doutorado) — Université Paris-Est, 2010.

27 ALVES, V. N. **Estudo de novas estratégias para identificação de danos estruturais a partir de dados vibracionais.** 188 p. Dissertação (Mestrado) — Universidade Federal de Ouro Preto, Ouro Preto, Brasil, 2012.

28 DOEBLING, S. W.; FARRAR, C. R.; PRIME, M. B.; SHEVITZ, D. W. Damage identification and health monitoring of structural and mechanical systems from changes in their vibration characteristics: a literature review. Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 1996.

29 PANDEY, A.; BISWAS, M.; SAMMAN, M. Damage detection from changes in curvature mode shapes. **Journal of sound and vibration**, Elsevier, v. 145, n. 2, p. 321–332, 1991.

30 KIM, J.; STUBBS, N. Assessment of the relative impact of model uncertainty on the accuracy of global nondestructive damage detection in structures. **Report prepared for New Mexico State University**, 1993.

31 PANDEY, A.; BISWAS, M. Damage detection in structures using changes in flexibility. **Journal of sound and vibration**, Elsevier, v. 169, n. 1, p. 3–17, 1994.

32 ALVANDI, A.; CREMONA, C. Assessment of vibration-based damage identification techniques. **Journal of sound and vibration**, Elsevier, v. 292, n. 1-2, p. 179–202, 2006.

33 CURY, A. A.; BORGES, C. C.; BARBOSA, F. S. A two-step technique for damage assessment using numerical and experimental vibration data. **Structural Health Monitoring**, SAGE Publications Sage UK: London, England, v. 10, n. 4, p. 417–428, 2011.

34 CHANG, C.; CHANG, T.; XU, Y.; WANG, M. Structural damage detection using an iterative neural network. **Journal of intelligent material systems and structures**, Sage Publications Sage CA: Thousand Oaks, CA, v. 11, n. 1, p. 32–42, 2000.

35 IWASAKI, A.; TODOROKI, A.; SHIMAMURA, Y.; KOBAYASHI, H. An unsupervised statistical damage detection method for structural health monitoring (applied to detection of delamination of a composite beam). **Smart materials and structures**, IOP Publishing, v. 13, n. 5, p. N80, 2004.

36 ALVES, V.; CURY, A.; ROITMAN, N.; MAGLUTA, C.; CREMONA, C. Novelty detection for shm using raw acceleration measurements. **Structural Control and Health Monitoring**, Wiley Online Library, v. 22, n. 9, p. 1193–1207, 2015.

37 MA, X.; LIN, Y.; NIE, Z.; MA, H. Structural damage identification based on unsupervised feature-extraction via variational auto-encoder. **Measurement**, Elsevier, v. 160, p. 107811, 2020.

38 POLLASTRO, A.; TESTA, G.; BILOTTA, A.; PREVETE, R. Semi-supervised detection of structural damage using variational autoencoder and a one-class support vector machine. **IEEE Access**, IEEE, 2023.

39 MA, X.; LIN, Y.; NIE, Z.; MA, H. Structural damage identification based on unsupervised feature-extraction via variational auto-encoder. **Measurement**, Elsevier Ltd, v. 160, p. 1–15, 2020.

40 ANAISSI, A.; ZANDAVI, S. M.; SULEIMAN, B.; NAJI, M.; BRAYTEE, A. Multi-objective variational autoencoder: an application for smart infrastructure maintenance. **Appl Intell**, Applied Intelligence, v. 53, p. 12047–12062, 2023.

41 TERBUCH, A.; O'LEARY, P.; KHALILI-MOTLAGH-KASMAEI, N.; AUER, P.; ZÖHRER, A.; WINTER, V. Detecting anomalous multivariate time-series via hybrid machine learning. **IEEE transactions on instrumentation and measurement**, IEEE, v. 72, p. 1–11, 2023.

42 PRINCIPE, J. C.; EULIANO, N. R.; LEFEBVRE, W. C. **Neural and Adaptive Systems: Fundamentals Through Simulations**. [S.l.]: John Wiley and Sons, 2000.

43 HAYKIN, S. **Neural networks: a comprehensive foundation**. [S.l.]: Pearson Education, 1999.

44 ABRAHAM, A. Meta learning evolutionary artificial neural networks. **Neurocomputing**, Elsevier Ltd, v. 56, p. 1–38, 2004.

45 BISHOP, C. M. **Pattern Recignition and Machine Learning**. Inglaterra: Springer-Verlag, 2006.

46 CYBENKO, G. **Approximation by Superpositions of a Sigmoidal Function**. [S.l.]: Mathematics of Control, Signal and Systems, 1989. 303-314 p.

47 MARMOLIN, H. Subjective mse measures. **IEEE transactions on systems, man, and cybernetics**, IEEE, v. 16, n. 3, p. 486–489, 1986.

48 MCLACHLAN, G. J. Mahalanobis distance. **Resonance**, v. 4, n. 6, p. 20–26, 1999.

49 FINOTTI, R. P.; SILVA, C. F. d.; OLIVEIRA, P. H. E.; BARBOSA, F. d. S.; CURY, A. A.; SILVA, R. C. Novelty detection on a laboratory benchmark slender structure using an unsupervised deep learning algorithm. **Latin American Journal of Solids and Structures**, SciELO Brasil, v. 20, p. e512, 2023.

50 ROECK, G. D.; PEETERS, B.; MAECK, J.; GUIDO. **Dynamic monitoring of civil engineering structures**. [S.l.]: eng, 2000.

51 MAECK, J.; PEETERS, B.; ROECK, G. D. Damage identification on the z24 bridge using vibration monitoring. **Smart materials and structures**, IOP Publishing, v. 10, n. 3, p. 512, 2001.

6 APÊNDICES

6.1 APÊNDICE A

Nessa seção, são apresentados os códigos utilizados na presente pesquisa, que foram feitos no software *Matlab R2023a*. De modo a simplificar o seu uso, foram criadas funções para cada etapa de funcionamento. Todo o material se encontra nesse drive.

6.1.1 Código do pórtico bi-engastado

```
    %% 1) INICIALIZAÇÃO
clear, clc, close all

%Inserir o caminho até a pasta de ENSAIOS
caminho='C:\Users\Universidade Federal\Desktop\Luiz\Z24\Dados';

%Definição dos parâmetros inciais
n_tr=300; %Número de dados para treino (para cada estágio dano utilizado)
n_test=468; %Número de dados para teste(para cada estágio dano utilizado)
num_col = 9; % Número de colisões em cada estágio de dano
num_points=65530; %Número de pontos pegados por ensaio
T=10; %Tempo de aquisição do sinal
%(caso seja escolhido o domínio da frequência)
hz=63; %Tamanho em hz do sinal
%(caso seja domínio da frequência)
num_ac=1; %Número do acelerômetro analisado
N_latent=100; %Espaço latente

%Configuração da arquitetura da RNA
TipoAE='VAE'; %Tipo de autocodificador utilizado
%(pode ser 'VAE' ou 'AE')
CamadaCod={'Bi-LSTM'}; %Tipo de camada codificadora
```

```
%('FC','LSTM' ou 'Bi-LSTM')
CamadaDec={'Bi-LSTM'}; %Tipo de camada codificadora
%('FC','LSTM' ou 'Bi-LSTM')

%Intervalo de otimização para os parâmetros
NeuCod=[500 600]; %Intervalo de neurônios para otimização
%na camada codificadora
NeuDec= [500 600];%Intervalo de neurônios para otimização
%na camada decodificadora
NumEpoc=[1 100]; %Intervalo de épocas a serem otimizadas
ger=10; %Número de gerações para otimizar os parâmetros do AE
pop=10; %Número de população para otimizar os parâmetros do AE

%Número de testes para classificação
N=10;

%% 2) Carregamento de dados
%Descomentar aquele a ser utilizado
% 2.1 - Sinais no domínio do tempo
[TrainData,ValidationData,TestData,OptimizationData]=
CarregaSinais_Z24_Tempo(num_col,num_points,num_ac,n_tr,caminho);

% 2.2 - Sinais no domínio da frequência
% [TrainData,ValidationData,TestData,OptimizationData]=
CarregaSinais_Z24_Frequencia(num_col,num_points,num_ac,n_tr,caminho,T,hz);

% 2.3 - Sinais no domínio do tempo somados com o da frequência
% [TrainData,ValidationData,TestData,OptimizationData]=
CarregaSinais_Z24_FrequenciaETempo(hz,T,num_col,num_points,num_ac,n_tr,caminho);

clear n_val num_col num_points

%Caso não queira rodar a otimização, inserir os parâmetros da arquitetura
```

```
%da rede nesse local.
% hpObj=HyperparametersAED();
% hpObj.setHyperparametersAED('AutoencoderType',TipoAE,'LayersEncoder', ...
%     CamadaCod,'NeuronsEncoder',1055, ...
%     'LayersDecoder',CamadaDec,'NeuronsDecoder',1042,...
%     'WeightingKL',5, ...
%     'LearningRate',9390*10^(-7), ...
%     'MiniBatchSize',28, ...
%     'NumberEpoch',95,...
%     'LatentDim',N_latent)

%% 2) Otimização
[hpObj]=Otimizacao(ger,pop,TipoAE,CamadaCod,CamadaDec,...
N_latent,NeuCod,NeuDec,NumEpoc,OptimizationData);

clear OptimizationData ger pop NeuCod TipoAE CamadaCod
clear CamadaDec N_latent NeuDec NumEpoc

%% 3) Classificação dos dados
%Descomentar aquela métrica a ser utilizada
% 3.1 - MSE com a Distância de Mahalanobis
[acertosD0,errosD0,errosDanos,acertosDanos] =
Classificacao_Z24_MSE_Mahalanobis(N,hpObj,TrainData,ValidationData,...
TestData,n_test,n_tr,num_ac);

% 3.2 - MSE
% [acertosD0,errosD0,errosDanos,acertosDanos] =
Classificacao_Z24_MSE(N,hpObj,TrainData,ValidationData,TestData,...
n_test,n_tr,num_ac);
```

6.1.2 Código da Ponte Z24

```
%% 1) INICIALIZAÇÃO
clear, clc, close all

%Inserir o caminho até a pasta de ENSAIOS
caminho='C:\Users\Universidade Federal\Desktop\Luiz\Z24\Dados';

%Definição dos parâmetros inciais
n_tr=300; %Número de dados para treino (para cada estágio dano utilizado)
n_test=468; %Número de dados para teste (para cada estágio dano utilizado)
num_col = 9; % Número de colisões em cada estágio de dano
num_points=65530; %Número de pontos pegados por ensaio
T=10; %Tempo de aquisição do sinal
%(caso seja escolhido o domínio da frequência)
hz=63; %Tamanho em hz do sinal (caso seja domínio da frequência)
num_ac=1; %Número do acelerômetro analisado
N_latent=100; %Espaço latente

%Configuração da arquitetura da RNA
TipoAE='VAE'; %Tipo de autocodificador utilizado
%(pode ser 'VAE' ou 'AE')
CamadaCod={'Bi-LSTM'}; %Tipo de camada codificadora
%('FC','LSTM' ou 'Bi-LSTM')
CamadaDec={'Bi-LSTM'}; %Tipo de camada codificadora
%('FC','LSTM' ou 'Bi-LSTM')

%Intervalo de otimização para os parâmetros
NeuCod=[500 600]; %Intervalo de neurônios para otimização
%na camada codificadora
NeuDec= [500 600];%Intervalo de neurônios para otimização
%na camada decodificadora
NumEpoc=[1 100]; %Intervalo de épocas a serem otimizadas
```

```
ger=10; %Número de gerações para otimizar os parâmetros do AE
pop=10; %Número de população para otimizar os parâmetros do AE

%Número de testes para classificação
N=10;

%% 2) Carregamento de dados
%Descomentar aquele a ser utilizado
% 2.1 - Sinais no domínio do tempo
[TrainData,ValidationData,TestData,OptimizationData]=
CarregaSinais_Z24_Tempo(num_col,num_points,num_ac,n_tr,caminho);

% 2.2 - Sinais no domínio da frequência
% [TrainData,ValidationData,TestData,OptimizationData]=
CarregaSinais_Z24_Frequencia(num_col,num_points,num_ac,n_tr,caminho,T,hz);

% 2.3 - Sinais no domínio do tempo somados com o da frequência
% [TrainData,ValidationData,TestData,OptimizationData]=
CarregaSinais_Z24_FrequenciaETempo(hz,T,num_col,num_points,num_ac,...
n_tr,caminho);

clear n_val num_col num_points

%Caso não queira rodar a otimização, inserir os parâmetros da arquitetura
%da rede nesse local.
% hpObj=HyperparametersAED();
% hpObj.setHyperparametersAED('AutoencoderType',TipoAE,'LayersEncoder', ...
%     CamadaCod,'NeuronsEncoder',1055, ...
%     'LayersDecoder',CamadaDec,'NeuronsDecoder',1042,...
%     'WeightingKL',5, ...
%     'LearningRate',9390*10^(-7), ...
%     'MiniBatchSize',28, ...
%     'NumberEpoch',95,...
```

```
%     'LatentDim',N_latent)

%% 2) Otimização
[hpObj]=Otimizacao(ger,pop,TipoAE,CamadaCod,CamadaDec,N_latent,NeuCod,...
NeuDec,NumEpoc,OptimizationData);

clear OptimizationData ger pop NeuCod TipoAE CamadaCod
clear CamadaDec N_latent NeuDec NumEpoc

%% 3) Classificação dos dados
%Descomentar aquela métrica a ser utilizada
% 3.1 - MSE com a Distância de Mahalanobis
[acertosD0,errosD0,errosDanos,acertosDanos] =
Classificacao_Z24_MSE_Mahalanobis(N,hpObj,TrainData,ValidationData,...
TestData,n_test,n_tr,num_ac);

% 3.2 - MSE
% [acertosD0,errosD0,errosDanos,acertosDanos] =
Classificacao_Z24_MSE(N,hpObj,TrainData,ValidationData,TestData,...
n_test,n_tr,num_ac);
```

6.2 APÊNDICE B

6.2.1 Fase 1

Após o insucesso de um estudo avançado com um tipo de Rede Neural Artificial que utiliza Decomposição em Modo Dinâmico (DMD, do inglês *dynamic mode decomposition*) para aplicação na detecção de danos estruturais, o estudo dessa pesquisa se voltou para o material apresentado por Terbuch [17] em que os resultados se mostraram mais condizentes para tal tarefa.

Após uma revisão bibliográfica e um estudo inicial mais avançado, alguns

testes foram feitos com dois modelos diferentes utilizando três dos cinco níveis de danos ensaiados nessa estrutura. Esses dois modelos diferenciam-se da seguinte forma:

1. **Modelo 1:** Durante a fase de treinamento foram usados parte dos dados do cenário 1. Na fase de validação foram usados os dados cenário 1 (dano 0) que não foram usados na fase de treinamento. Já para a fase de teste, utilizou-se os dados do cenário 2 (dano 1);

2. **Modelo 2:** Durante a fase de treinamento foram usados parte dos dados dos cenários 1 e 2 (danos 0 e 1). Na fase de validação foram usados os dados dos cenários 1 e 2 que não foram usados na fase de treinamento. Já para a fase de teste, utilizou-se os dados do cenário 3 (dano 2).

Em ambos estudos a matriz de entrada no algoritmo é a mesma, em cada acelerômetro são 300 ensaios contendo 2000 pontos, por cenário. Desse modo, durante a fase de treinamento no modelo 1 são pegados 200 ensaios do cenário 1 para treino e os outros 100 são usados para validação; e durante a fase de teste, todos os 300 ensaios do cenário 2 são utilizados. Nessa mesma analogia, durante a fase de treinamento no modelo 2 são pegados 200 ensaios do cenário 1 e 200 do cenário 2 para treino e os outros 100 de cada um desses são usados para validação; já durante a fase de teste, todos os 300 ensaios do cenário 3 são utilizados.

Além disso, a RNA utilizada possuía seus neurônios todos conectados com a camada subsequente. Ou seja, as funções usadas nos testes do algoritmo tanto para codificar, quanto para decodificar, é a FC. Ademais, o número de neurônios utilizado nessas duas camadas foi 20, o qual foi feito de modo empírico. Por fim, cabe ressaltar que o Autocodificador usado nos resultados que se seguem é o variacional com a dimensão do espaço latente igual a dois. Os demais parâmetros utilizados para cada modelo encontram-se nas Tabelas 5 e 6.

Tabela 5 – Tabela com os parâmetros utilizados no Modelo 1

Parâmetro	**Significado**	**Atributo**
AutoencoderType	Tipo de autocodificador	VAE
LayersEncoder	Camadas de codificação	FC
LayersDecoder	Camadas de decodificação	FC
NeuronsEncoder	Neurônios de codificação	20
NeuronsDecoder	Neurônios de decodificação	20
LatentDim	Dimensão do Espaço Latente	2
NumberEpoch	Número de Épocas	30
NumberFeature	Número de Características	1
LearningRate	Taxa de aprendizado	0.05
MiniBatchSize	Tamanho do lote utilizado	15
ExecutionEnvironment	Onde ocorrerá a execução	'auto'
WeightingKL	Divergência de Kullback-Leibler	1
OutputTransferFunction	Intervalo de saída do Autoencoder	'none'

Fonte: Autor

Tabela 6 – Tabela com os parâmetros utilizados no Modelo 2

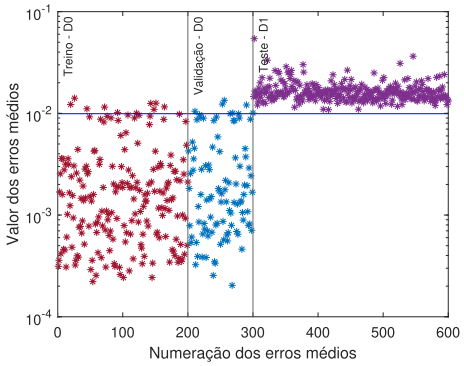
Parâmetro	**Significado**	**Atributo**
AutoencoderType	Tipo de autocodificador	VAE
LayersEncoder	Camadas de codificação	FC
LayersDecoder	Camadas de decodificação	FC
NeuronsEncoder	Neurônios de codificação	20
NeuronsDecoder	Neurônios de decodificação	20
LatentDim	Dimensão do Espaço Latente	2
NumberEpoch	Número de Épocas	80
NumberFeature	Número de Características	1
LearningRate	Taxa de aprendizado	0.05
MiniBatchSize	Tamanho do lote utilizado	15
ExecutionEnvironment	Onde ocorrerá a execução	'auto'
WeightingKL	Divergência de Kullback-Leibler	1
OutputTransferFunction	Intervalo de saída do Autoencoder	'none'

Fonte: Autor

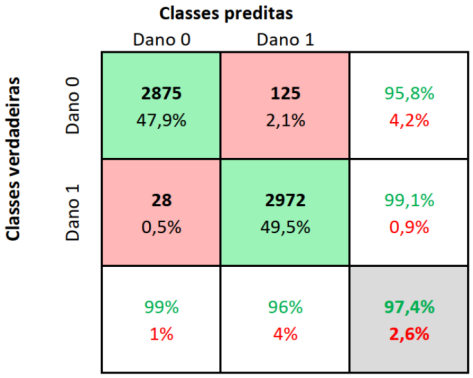
Assim, foram inseridos os dados na rede neural e o algoritmo retornou um sinal reconstruído. Após essa etapa, para cada ensaio, foi calculado o erro

através do MSE. Nesse sentido, a última etapa foi a classificação desses erros, com a finalidade de conhecer a capacidade do algoritmo de distinguir os diferentes cenários de danos. A métrica adotada foi a limiar de 95% dos erros médios obtidos da fase de treinamento do código.

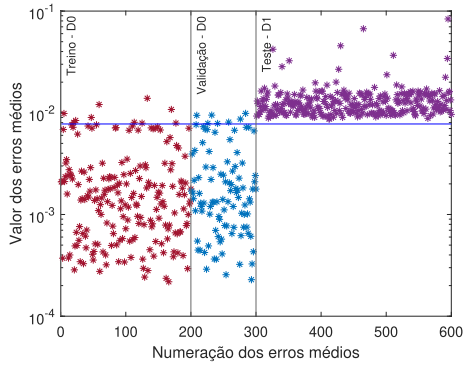
Dito isso, os resultados encontrados foram plotados e estão nas Figuras 38 e 39. Nelas é possível observar os erros médios obtidos de cada um dos 4 acelerômetros, por ensaio do algoritmo. Na Figura 38 os resultados são de acordo com o modelo 1, já na Figura 39 de acordo com o modelo 2.



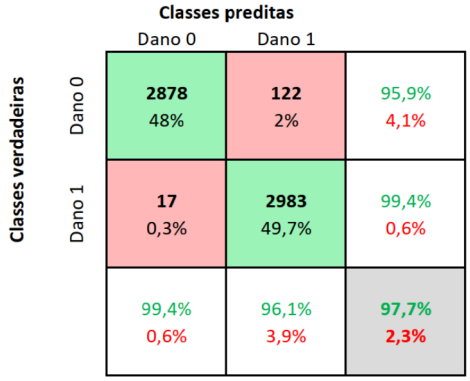
(a) AC1 - Resultado típico



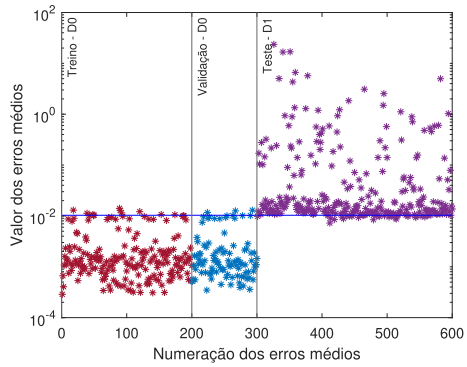
(b) AC 1 - Matriz de confusão



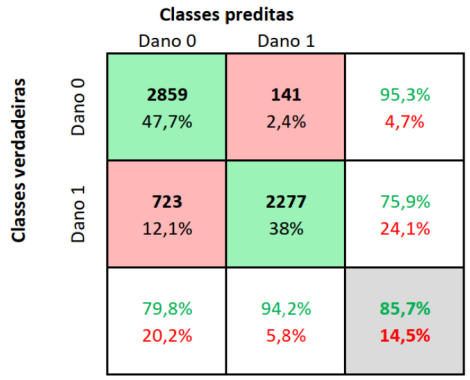
(c) AC2 - Resultado típico



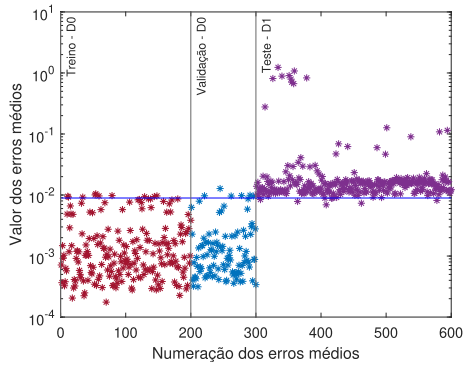
(d) AC 2 - Matriz de confusão



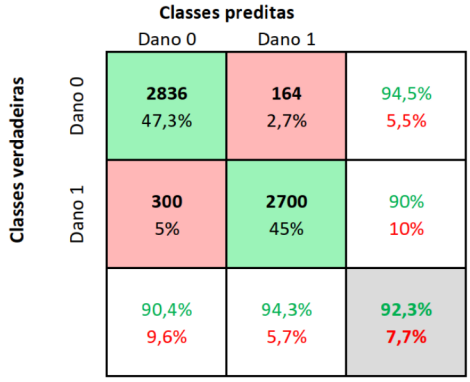
(e) AC3 - Resultado típico



(f) AC 3 - Matriz de confusão



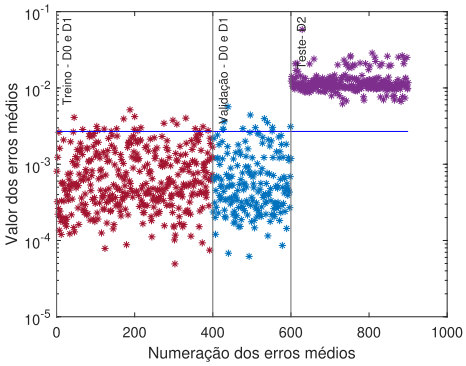
(g) AC4 - Resultado típico



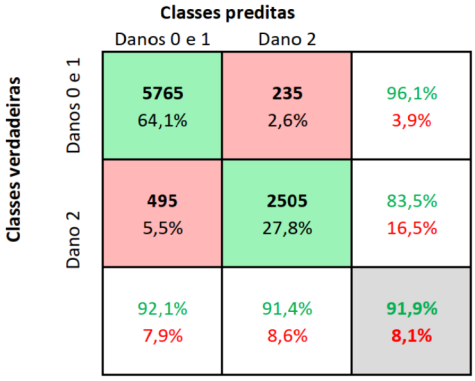
(h) AC 4 - Matriz de confusão

Figura 38 – Resultados Modelo 1

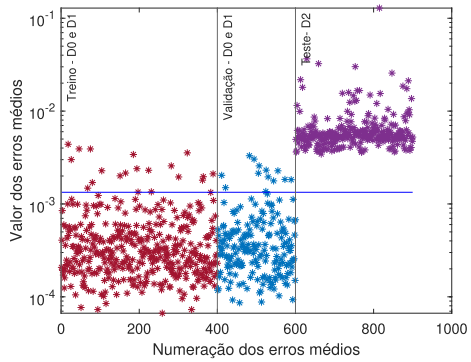
Fonte: Autor



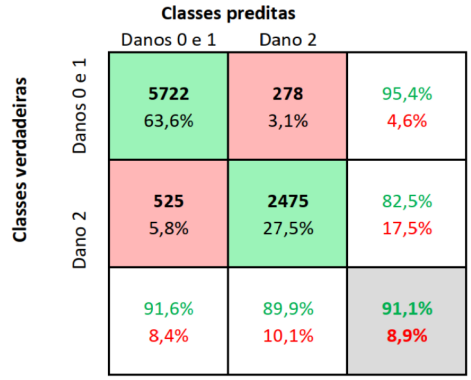
(a) AC1 - Resultado típico



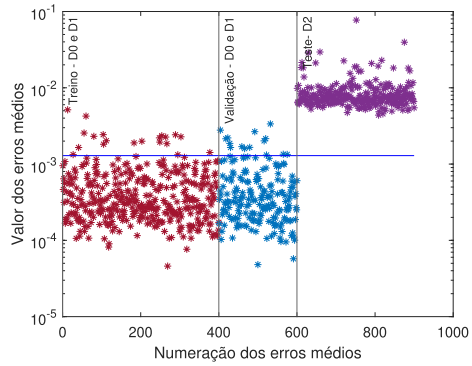
(b) AC 1 - Matriz de confusão



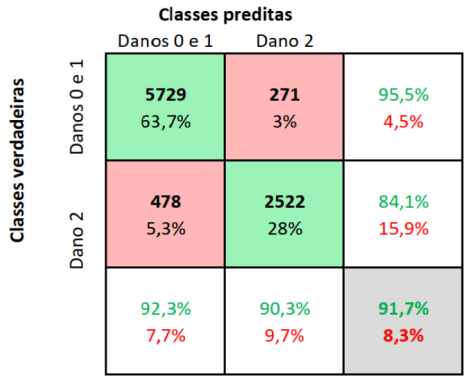
(c) AC2 - Resultado típico



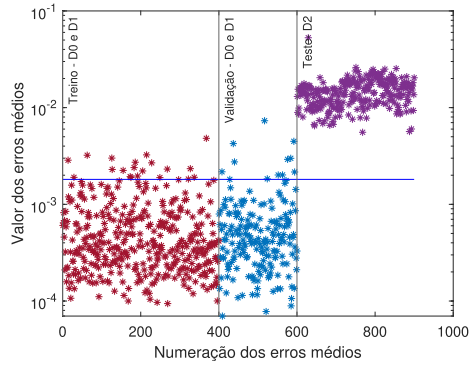
(d) AC 2 - Matriz de confusão



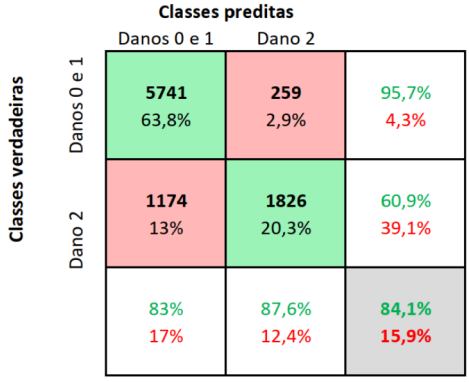
(e) AC3 - Resultado típico



(f) AC 3 - Matriz de confusão



(g) AC4 - Resultado típico



(h) AC 4 - Matriz de confusão

Figura 39 – Resultados Modelo 2

Fonte: Autor

Nos dois modelos é possível ver uma boa tendência de separação dos dados de treinamento e validação para os de teste em relação ao limiar (reta em cor azul). De modo a interpretar melhor esses dados, foram realizados dez testes com a mesma configuração da RNA para cada acelerômetro, por modelo.

Aqui foi possível observar que a técnica a ser investigada possui bons resultados iniciais de maneira geral. Tal afirmação é corroborada pelos resultados obtidos nos modelos, haja vista que o percentil mínimo de acerto está perto dos 84%. Para cada modelo, o tempo médio é de 15 minutos rodando em um bom computador (o utilizado foi de Intel Core i7-6700K CPU 2 núcleos 4.00 GHz e 16 GB de RAM).

Entretanto, nesse modelo foram variados todos os parâmetros de forma manual e empírica. Quase 1.000 modelos com arquiteturas diferentes foram criados. No entanto, ainda era necessário variar melhor e de maneira automática os parâmetros. Contudo, dessa primeira fase foi extraída a limiar de 95% que classifica os resultados das funções erros para o resultado final da presente pesquisa.

6.2.2 Fase 2

Nessa etapa da pesquisa foi publicado o estudo de Terbuch *et al.* [41]. Assim, alguns dos parâmetros puderam ser otimizados, melhorando consideravelmente os resultados. Foram testadas novas redes (agora com os cinco danos), com novos números de camadas latentes e novas funções erros (ORSR, Distância de Mahalanobis e Resíduo), conforme mostrado na Figura 40.

PÓRTICO					
ARQUITETURAS OTIMIZADAS					
Parâmetro	**Modelo 1**	**Modelo 2**	**Modelo 3**	**Modelo 4**	**Modelo 5**
AutoencoderType	VAE	VAE	VAE	VAE	VAE
LayersEncoder	FC	FC	FC	FC	FC
LayersDecoder	FC	FC	FC	FC	FC
NeuronsEncoder	23	93	81	3	23
NeuronsDecoder	79	65	22	45	79
LatentDim	25	50	75	100	25
NumberEpoch	87	88	99	94	87
NumberFeature	2000	2000	2000	2000	2000
LearningRate	0,0008771	0,0008763	0,0006297	0,0008013	0,0008771
MiniBatchSize	14	18	49	2	14
ExecutionEnvironment	AUTO	AUTO	AUTO	AUTO	AUTO
WeightingKL	0	0	3	91	0
OutputTransferFunction	NONE	NONE	NONE	NONE	NONE
Função erro	MSE	MSE	MSE	MSE	RESÍDUO COM MAHALANOBIS
Parâmetro	**Modelo 6**	**Modelo 7**	**Modelo 8**	**Modelo 9**	**Modelo 10**
AutoencoderType	VAE	VAE	VAE	VAE	VAE
LayersEncoder	FC	FC	FC	FC	FC
LayersDecoder	FC	FC	FC	FC	FC
NeuronsEncoder	23	81	3	23	23
NeuronsDecoder	79	22	45	79	79
LatentDim	25	75	100	25	25
NumberEpoch	87	99	94	87	87
NumberFeature	2000	2000	2000	2000	2000
LearningRate	0,0008771	0,0006297	0,0008013	0,0008771	0,0008771
MiniBatchSize	14	49	2	14	14
ExecutionEnvironment	AUTO	AUTO	AUTO	AUTO	AUTO
WeightingKL	0	3	91	0	0
OutputTransferFunction	NONE	NONE	NONE	NONE	NONE
Função erro	MSE+MAHALANOBIS	MSE+MAHALANOBIS	MSE+MAHALANOBIS	ORSR	ORSR+MAHALANOBIS

Figura 40 – Arquiteturas otimizadas - fase 2

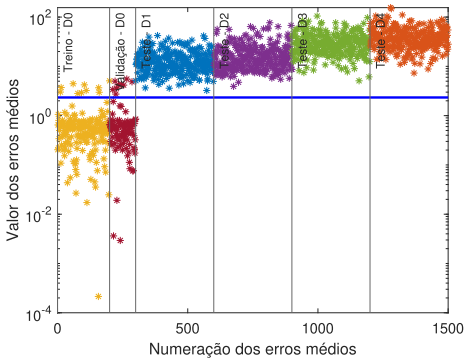
Fonte: Autor

Aqui, os parâmetros inseridos no algoritmo para otimização foram:

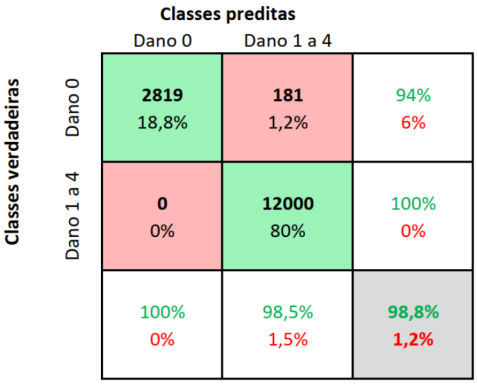
1. **Tipo de Autocodificador:** 'VAE';
2. **Camadas do codificador:** Camada única, FC;
3. **Camadas do decodificador:** Igual ao codificador;

4. **Neurônios do codificador:** O intervalo escolhido para otimização foi entre 0 e 100 neurônios;
5. **Neurônios do decodificador:** Igual ao codificador;
6. **Número de variáveis na camada latente:** Indicada na Figura 40 para cada um dos modelos;
7. **Número de épocas:** O intervalo escolhido para otimização foi entre 0 e 100 épocas;
8. **Número de características:** Número retornado pelo programa foi de 2000;
9. **Taxa de aprendizado:** O intervalo escolhido para otimização foi entre 0,00001 e 0,001;
10. **Tamanho do minilote:** O intervalo escolhido para otimização foi entre 2 e 100 para o tamanho do minilote;
11. **Local de execução do treinamento da rede neural:** Os computadores que foram utilizados nos testes, não possuíam placa de vídeo. Nesse caso, foi optado para colocar 'CPU' nos modelos;
12. **Divergência de Kullback-Leiber:** O intervalo escolhido para otimização foi entre 0 e 100;
13. **Função de saída dos dados:** Não se utilizou nenhuma função de saída para os modelos, portanto esse parâmetro foi fixado em 'none';

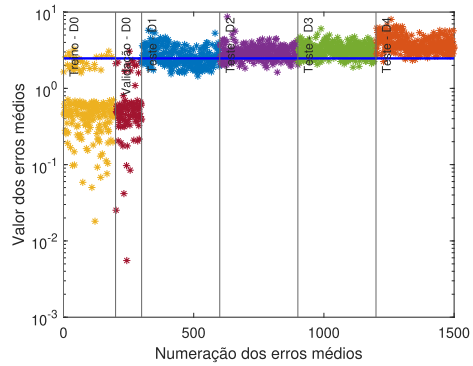
Aqui foi possível ver que para o pórtico as funções erros ideais entre as testadas foram a MSE e a junção de MSE com Mahalanobis. Além disso, foi visto que o melhor número de neurônios nas camadas latentes seriam os de 75 e 100. Todos os outros parâmetros de otimização foram mantidos para a versão final da pesquisa, com exceção do intervalo de neurônios e as funções camadas codificadoras e decodificadores. Os resultados das melhores arquiteturas estão dispostos na Figura 41, todos correspondem aos obtidos através dos dados do acelerômetro 3.



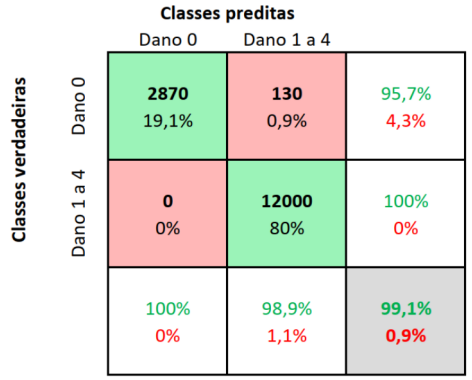
(a) Modelo 3 - Resultado típico



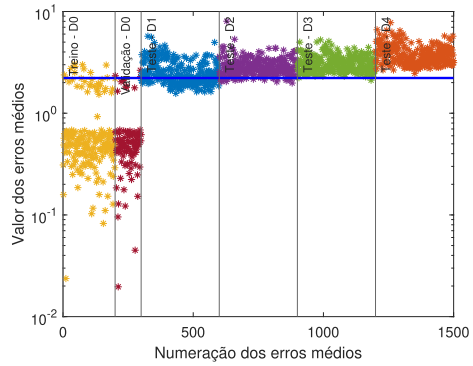
(b) Modelo 3 - Matriz de confusão



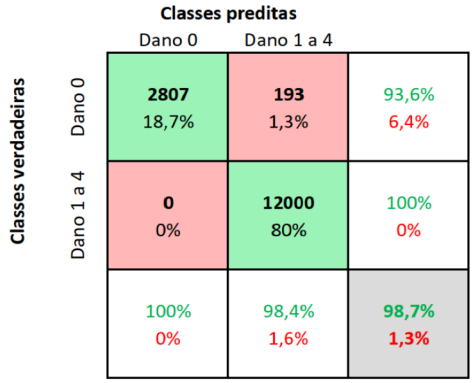
(c) Modelo 4 - Resultado típico



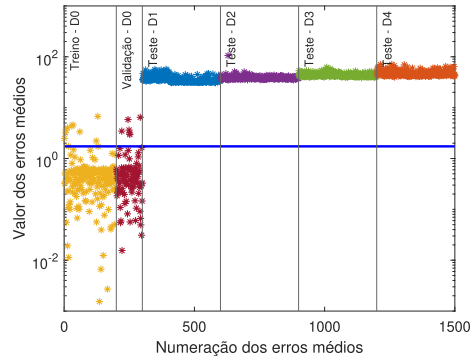
(d) Modelo 4 - Matriz de confusão



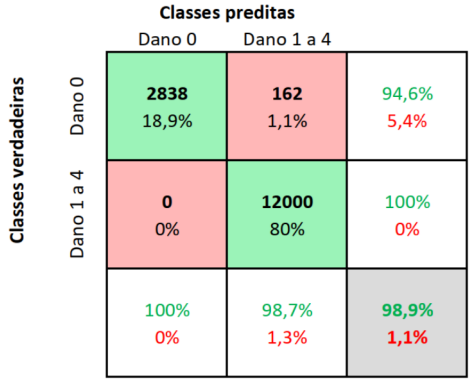
(e) Modelo 7 - Resultado típico



(f) Modelo 7 - Matriz de confusão



(g) Modelo 8 - Resultado típico



(h) Modelo 8 - Matriz de confusão

Figura 41 – Melhores resultados fase 2

Fonte: Autor

Nessa fase, foi optado apenas por otimizar os parâmetros em relação ao acelerômetro 3, haja vista que o tempo computacional era grande e ainda havia outros testes para melhorar a arquitetura. Aqui, os modelos testados possuem um tempo médio computacional entre 4 e 10 horas por modelo.

6.2.3 Fase 3

Após obter e entender os resultados das redes na fase 2, foi alterado o modo como se entrava com os dados da base de dados de entrada. Dessa forma, foram testadas redes que possuíam como entrada conjunto de dados nos domínios do tempo, da frequência e uma aglutinação de dados nesses dois domínios.

Além disso, em alguns casos foram inseridos diferentes intervalos para o número de neurônios tanto no codificador, quanto na decodificador. Como uma terceira alteração, foram testadas camadas codificadoras e decodificadoras simples e duplas com funções de ligações iguais ou diferentes (variando entre FC, LSTM e Bi-LSTM). Os resultados após otimização se encontram nas Figuras 42 e 43, todos esse modelos foram testados com o uso das métricas MSE e a junção de MSE com a Distância de Mahalanobis.

PÓRTICO				
PARÂMETROS OTIMIZADOS				
Parâmetro	**Modelo 1**	**Modelo 2**	**Modelo 3**	**Modelo 4**
AutoencoderType	VAE	VAE	VAE	VAE
LayersEncoder	FC	FC	FC	FC
LayersDecoder	FC	FC	FC	FC
NeuronsEncoder	23	93	81	3
NeuronsDecoder	79	65	22	45
LatentDim	25	50	75	100
NumberEpoch	87	88	99	94
NumberFeature	2000	2000	2000	2000
LearningRate	0,0008771	0,0008763	0,0006297	0,0008013
MiniBatchSize	14	18	49	2
ExecutionEnvironment	AUTO	AUTO	AUTO	AUTO
WeightingKL	0	0	3	91
OutputTransferFunction	NONE	NONE	NONE	NONE
Domínio	FREQUÊNCIA	FREQUÊNCIA	FREQUÊNCIA	FREQUÊNCIA
Parâmetro	**Modelo 5**	**Modelo 6**	**Modelo 7**	**Modelo 8**
AutoencoderType	VAE	VAE	VAE	VAE
LayersEncoder	FC	FC,FC	LSTM	LSTM,LSTM
LayersDecoder	FC	FC,FC	LSTM	LSTM,LSTM
NeuronsEncoder	1072	[553 1014]	1055	[15 1]
NeuronsDecoder	1050	[1037 553]	1042	[89 20]
LatentDim	75	75	100	100
NumberEpoch	56	82	95	95
NumberFeature	2000	2000	2000	2000
LearningRate	0,0005949	0,0008862	0,0006965	0,0007991
MiniBatchSize	81	13	55	15
ExecutionEnvironment	AUTO'	AUTO'	AUTO'	AUTO'
WeightingKL	6	1	5	72
OutputTransferFunction	NONE'	NONE'	NONE'	NONE'
Domínio	TEMPO	TEMPO	TEMPO	TEMPO

Figura 42 – Arquiteturas otimizadas - fase 3 - parte 1

Fonte: Autor

Parâmetro	Modelo 9	Modelo 10	Modelo 11	Modelo 12
AutoencoderType	VAE	VAE	VAE	VAE
LayersEncoder	Bi-LSTM	Bi-LSTM,Bi-LSTM	FC,LSTM	FC,LSTM
LayersDecoder	Bi-LSTM	Bi-LSTM,Bi-LSTM	FC,LSTM	FC,LSTM
NeuronsEncoder	1010	[506 1087]	[525 1084]	[554 1079]
NeuronsDecoder	1060	[1065 506]	[1081 555]	[1038 551]
LatentDim	100	100	100	100
NumberEpoch	94	82	70	77
NumberFeature	2000	2000	2000	261
LearningRate	0,0008939	0,0006746	0,0009982	0,0008009
MiniBatchSize	60	58	10	64
ExecutionEnvironment	AUTO'	AUTO'	AUTO'	AUTO'
WeightingKL	14	8	19	66
OutputTransferFunction	NONE'	NONE'	NONE'	NONE'
Domínio	TEMPO	TEMPO	TEMPO	FREQUÊNCIA

Parâmetro	Modelo 13	Modelo 14
AutoencoderType	VAE	VAE
LayersEncoder	FC,LSTM	FC,LSTM
LayersDecoder	FC,LSTM	FC,LSTM
NeuronsEncoder	[11 38]	[35 84]
NeuronsDecoder	[86 20]	[74 17]
LatentDim	100	100
NumberEpoch	34	95
NumberFeature	2000	2000
LearningRate	0,0009843	0,0009523
MiniBatchSize	18	7
ExecutionEnvironment	AUTO'	AUTO'
WeightingKL	1	45
OutputTransferFunction	NONE'	NONE'
Domínio	TEMPO	FREQUÊNCIA

Figura 43 – Arquiteturas otimizadas - fase 3 - parte 2

Fonte: Autor

Aqui, os parâmetros inseridos no algoritmo para otimização foram:

1. **Tipo de Autocodificador:** 'VAE';
2. **Camadas do codificador:** Camadas com novos tipos de ligações entre os neurônios FC, LSTM e Bi-LSTM. Além disso, foram testadas camadas com duas funções iguais ou misturadas entre si;
3. **Camadas do decodificador:** Igual ao codificador;

4. **Neurônios do codificador:** Os intervalos escolhidos para otimização variam entre: [0;100], [400;500], [1000;1100];
5. **Neurônios do decodificador:** Igual ao codificador;
6. **Número de variáveis na camada latente:** Indicado nas Figuras 42 e 43 para cada um dos modelos;
7. **Número de épocas:** O intervalo escolhido para otimização foi entre 0 e 100 épocas;
8. **Número de características:** Número retornado pelo programa foi de 2000 (quando os dados de entrada eram no domínio do tempo), 261 (quando os dados de entrada eram no domínio da frequência) e 2261 (quando os dados de entrada eram no domínio do tempo somado aos mesmos dados no domínio da frequência);
9. **Taxa de aprendizado:** O intervalo escolhido para otimização foi entre 0,00001 e 0,001;
10. **Tamanho do minilote:** O intervalo escolhido para otimização foi entre 2 e 100 para o tamanho do minilote;
11. **Local de execução do treinamento da rede neural:** Os computadores que foram utilizados nos testes, não possuíam placa de vídeo. Nesse caso, foi optado para colocar 'CPU' nos modelos;
12. **Divergência de Kullback-Leiber:** O intervalo escolhido para otimização foi entre 0 e 100;
13. **Função de saída dos dados:** Não se utilizou nenhuma função de saída para os modelos, portanto esse parâmetro foi fixado em 'none';

Como era esperado, os resultados melhoraram bastante em relação a fase 2. Com a alteração no domínio dos dados de entrada foi possível perceber variações entre os resultados finais que podem ser significativos a depender da estrutura analisada, no caso específico do porte houve pouca variação. Além disso, o uso

ligações entre os neurônios de maneiras mais complexas (como LSTM e Bi-LSTM) melhoraram os resultados finais.

A alteração crucial foi a do intervalo de otimização do número de neurônios, a qual passou de 0 a 100 para 1000 a 1100, no domínio do tempo. Esse valor foi alterado em razão da lógica de redução das características de entrada camada de entrada até chegar na camada latente; e de seu aumento da camada latente para a camada de saída. Desse modo, os resultados finais foram retirados nessa fase, o que nas Figuras 42 e 43 correspondem aos modelos 5, 7 e 9. Nessa etapa, o tempo estimado para rodar cada um dos modelos testados varia de 6 a 12 horas.

6.2.4 Fase 4

Testadas todas as arquiteturas da fase anterior, o próximo passo da presente pesquisa foi realizar os testes com os dados do clássico caso da Ponte Z24. Aprendendo com a fase 3, apenas foi testado outros intervalos para os números de neurônios. Os resultados após otimização se encontram nas Figuras 44, 45, 46 e 47, todos esse modelos foram testados com o uso das métricas MSE e a junção de MSE com a Distância de Mahalanobis.

PONTE Z24					
PARÂMETROS OTIMIZADOS					
Parâmetro	**Modelo 1**	**Modelo 2**	**Modelo 3**	**Modelo 4**	**Modelo 5**
AutoencoderType	VAE	VAE	VAE	VAE	VAE
LayersEncoder	FC,FC	FC,FC	FC,FC	FC,FC	FC,FC
LayersDecoder	FC,FC	FC,FC	FC,FC	FC,FC	FC,FC
NeuronsEncoder	[566 586]	[356 309]	[1080 1028]	[554 576]	[350 370]
NeuronsDecoder	[584 566]	[305 320]	[1073 1013]	[566 516]	[306 370]
LatentDim	100	100	100	75	75
NumberEpoch	31	78	64	63	91
NumberFeature	1260	631	1891	1260	631
LearningRate	0,0009094	0,0009183	0,000919	0,0003181	0,0008196
MiniBatchSize	74	16	40	82	2
ExecutionEnvironment	AUTO	AUTO	AUTO	AUTO	AUTO
WeightingKL	96	19	47	0	63
OutputTransferFunction	NONE	NONE	NONE	NONE	NONE
Domínio	TEMPO	FREQUÊNCIA	TEMPO E FREQUÊNCIA	TEMPO	FREQUÊNCIA
Parâmetro	**Modelo 6**	**Modelo 7**	**Modelo 8**	**Modelo 9**	**Modelo 10**
AutoencoderType	VAE	VAE	VAE	VAE	VAE
LayersEncoder	FC,FC	FC	FC	FC	FC
LayersDecoder	FC,FC	FC	FC	FC	FC
NeuronsEncoder	[1066 1054]	563	341	1029	533
NeuronsDecoder	[1051 1005]	573	366	1048	517
LatentDim	75	10	10	10	2
NumberEpoch	66	81	96	67	58
NumberFeature	1891	1260	631	1891	1260
LearningRate	0,0009221	0,0008802	0,0007494	0,0007498	0,0003469
MiniBatchSize	22	28	66	18	22
ExecutionEnvironment	AUTO	AUTO	AUTO	AUTO	AUTO
WeightingKL	95	28	17	18	39
OutputTransferFunction	NONE	NONE	NONE	NONE	NONE
Domínio	TEMPO E FREQUÊNCIA	TEMPO	FREQUÊNCIA	TEMPO E FREQUÊNCIA	TEMPO

Figura 44 – Arquiteturas otimizadas - fase 4 - parte 1

Fonte: Autor

Parâmetro	Modelo 11	Modelo 12	Modelo 13	Modelo 14	Modelo 15
AutoencoderType	VAE	VAE	VAE	VAE	VAE
LayersEncoder	FC	FC	LSTM	LSTM	LSTM
LayersDecoder	FC	FC	LSTM	LSTM	LSTM
NeuronsEncoder	361	1019	519	312	1035
NeuronsDecoder	352	1078	512	312	1010
LatentDim	2	2	100	100	100
NumberEpoch	87	100	92	86	56
NumberFeature	631	1891	1260	631	1891
LearningRate	0,0002975	0,0006509	0,0008604	0,0008443	0,0008669
MiniBatchSize	3	28	18	2	49
ExecutionEnvironment	AUTO	AUTO	AUTO	AUTO	AUTO
WeightingKL	44	24	99	36	2
OutputTransferFunction	NONE	NONE	NONE	NONE	NONE
Domínio	FREQUENCIA	TEMPO E FREQUENCIA	TEMPO	FREQUENCIA	TEMPO E FREQUENCIA
Parâmetro	**Modelo 16**	**Modelo 17**	**Modelo 18**	**Modelo 19**	**Modelo 20**
AutoencoderType	VAE	VAE	VAE	VAE	VAE
LayersEncoder	LSTM	LSTM	LSTM	Bi-LSTM	Bi-LSTM
LayersDecoder	LSTM	LSTM	LSTM	Bi-LSTM	Bi-LSTM
NeuronsEncoder	565	372	1055	552	552
NeuronsDecoder	529	308	1072	512	512
LatentDim	75	75	75	100	100
NumberEpoch	91	100	45	73	73
NumberFeature	1260	631	1891	1260	1260
LearningRate	0,0008133	0,0008925	0,0008234	0,000518	0,000518
MiniBatchSize	6	65	3	100	100
ExecutionEnvironment	AUTO	AUTO	AUTO	AUTO	AUTO
WeightingKL	99	77	9	3	3
OutputTransferFunction	NONE	NONE	NONE	NONE	NONE
Domínio	TEMPO	FREQUENCIA	TEMPO E FREQUENCIA	TEMPO	FREQUENCIA

Figura 45 – Arquiteturas otimizadas - fase 4 - parte 2

Fonte: Autor

Parâmetro	Modelo 21	Modelo 22	Modelo 23	Modelo 24	Modelo 25
AutoencoderType	VAE	VAE	VAE	VAE	VAE
LayersEncoder	Bi-LSTM	Bi-LSTM	BI-LSTM	BI-LSTM	BI-LSTM
LayersDecoder	Bi-LSTM	Bi-LSTM	BI-LSTM	BI-LSTM	BI-LSTM
NeuronsEncoder	1023	533	309	1017	572
NeuronsDecoder	1046	539	308	1048	522
LatentDim	100	75	75	75	30
NumberEpoch	87	80	51	64	89
NumberFeature	1891	1260	631	1891	1260
LearningRate	0,000973	0,0008346	0,0009994	0,0009817	0,0005477
MiniBatchSize	22	10	3	21	19
ExecutionEnvironment	AUTO	AUTO	AUTO	AUTO	AUTO
WeightingKL	10	68	92	14	96
OutputTransferFunction	NONE	NONE	NONE	NONE	NONE
Domínio	TEMPO E FREQUÊNCIA	TEMPO	FREQUÊNCIA	TEMPO E FREQUÊNCIA	TEMPO
Parâmetro	**Modelo 26**	**Modelo 27**	**Modelo 28**	**Modelo 29**	**Modelo 30**
AutoencoderType	VAE	VAE	VAE	VAE	VAE
LayersEncoder	Bi-LSTM	Bi-LSTM	Bi-LSTM	Bi-LSTM	Bi-LSTM
LayersDecoder	Bi-LSTM	Bi-LSTM	Bi-LSTM	Bi-LSTM	Bi-LSTM
NeuronsEncoder	304	1054	554	336	1043
NeuronsDecoder	358	1059	516	350	1027
LatentDim	30	30	2	2	2
NumberEpoch	70	95	51	92	71
NumberFeature	631	1891	1260	631	1891
LearningRate	0,0006311	0,0009694	0,0001181	0,0003774	0,0001147
MiniBatchSize	62	20	16	46	12
ExecutionEnvironment	AUTO	AUTO	AUTO	AUTO	AUTO
WeightingKL	13	3	79	7	65
OutputTransferFunction	NONE	NONE	NONE	NONE	NONE
Domínio	FREQUÊNCIA	FREQUENCIA E TEMPO	TEMPO	FREQUÊNCIA	TEMPO E FREQUÊNCIA

Figura 46 – Arquiteturas otimizadas - fase 4 - parte 3

Fonte: Autor

Parâmetro	Modelo 31	Modelo 32	Modelo 33	Modelo 34	Modelo 35
AutoencoderType	VAE	VAE	VAE	VAE	VAE
LayersEncoder	FC	FC	FC	FC	FC
LayersDecoder	FC	FC	FC	FC	FC
NeuronsEncoder	547	551	368	360	560
NeuronsDecoder	553	522	356	369	573
LatentDim	75	100	75	100	75
NumberEpoch	96	78	82	74	26
NumberFeature	1260	1260	631	631	1891
LearningRate	0,0009025	0,0009427	0,0006732	0,0007482	0,0009718
MiniBatchSize	65	54	65	63	85
ExecutionEnvironment	AUTO	AUTO	AUTO	AUTO	AUTO
WeightingKL	79	57	9	1	0
OutputTransferFunction	NONE	NONE	NONE	NONE	NONE
Domínio	Tempo	TEMPO	FREQUÊNCIA	FREQUÊNCIA	FREQUENCIA E TEMPO

Parâmetro	Modelo 36
AutoencoderType	VAE
LayersEncoder	FC
LayersDecoder	FC
NeuronsEncoder	533
NeuronsDecoder	569
LatentDim	100
NumberEpoch	98
NumberFeature	1891
LearningRate	0,0009499
MiniBatchSize	48
ExecutionEnvironment	AUTO
WeightingKL	66
OutputTransferFunction	NONE
Domínio	FREQUENCIA E TEMPO

Figura 47 – Arquiteturas otimizadas - fase 4 - parte 4

Fonte: Autor

Aqui, os parâmetros inseridos no algoritmo para otimização foram:

1. **Tipo de Autocodificador:** 'VAE';
2. **Camadas do codificador:** Camadas com tipos de ligações entre os neurônios FC, LSTM e Bi-LSTM. Além disso, foram testadas camadas com duas funções iguais ou misturadas entre si;
3. **Camadas do decodificador:** Igual ao codificador;

4. **Neurônios do codificador:** Os intervalos escolhidos para otimização variam entre: [300;400], [500;600], [1000;1100];
5. **Neurônios do decodificador:** Igual ao codificador;
6. **Número de variáveis na camada latente:** Indicado nas Figuras 42 e 43 para cada um dos modelos;
7. **Número de épocas:** O intervalo escolhido para otimização foi entre 0 e 100 épocas;
8. **Número de características:** Número retornado pelo programa foi de 1260 (quando os dados de entrada eram no domínio do tempo), 631 (quando os dados de entrada eram no domínio da frequência) e 1891 (quando os dados de entrada eram no domínio do tempo somado aos mesmos dados no domínio da frequência);
9. **Taxa de aprendizado:** O intervalo escolhido para otimização foi entre 0,00001 e 0,001;
10. **Tamanho do minilote:** O intervalo escolhido para otimização foi entre 2 e 100 para o tamanho do minilote;
11. **Local de execução do treinamento da rede neural:** Os computadores que foram utilizados nos testes, não possuíam placa de vídeo. Nesse caso, foi optado para colocar 'None' nos modelos;
12. **Divergência de Kullback-Leiber:** O intervalo escolhido para otimização foi entre 0 e 100;
13. **Função de saída dos dados:** Não se utilizou nenhuma função de saída para os modelos, portanto esse parâmetro foi fixado em 'none';

Nessa fase foram obtidos os melhores resultados de testes para a Z24. Aqui, ao se variar os dados de entrada entre o domínio do tempo e domínio da frequência foram obtidos variações positivas nos resultados finais de classificação. Além disso, o uso de mais de uma camada na codificação e decodificação auxiliaram na melhoria de detecção de similaridades nos dados.

Desse modo, os resultados finais para a Z24 foram retirados nessa fase, o que nas Figuras 44 e 45 correspondem aos modelos 5, 14 e 20. Nessa etapa, o tempo estimado para rodar cada um dos modelos testados varia de 10 a 16 horas.