



**Politécnico
Castelo Branco**

Escola Superior
de Tecnologia

Manutenção Preditiva Baseada em Data Analytics e Machine Learning em Edge Computing

Rodrigo Filipe Correia Ribeiro

Nº 20181095

Orientadores

Professor Doutor Pedro Miguel Baptista Torres

Trabalho de Projeto apresentado à Escola Superior de Tecnologia do Instituto Politécnico de Castelo Branco para cumprimento dos requisitos necessários à obtenção do grau de Licenciado em Engenharia e Gestão Industrial, realizado sob a orientação científica do Professor Doutor Pedro Torres, do Instituto Politécnico de Castelo Branco.

Julho 2026

Composição do júri

Presidente do júri

Professor Fernando Júlio Marques Miranda, Professor Adjunto
Escola Superior de Tecnologia do Instituto Politécnico de Castelo Branco

Arguente

Professor Doutor Armando Lopes Ramalho, Professor Coordenador
Escola Superior de Tecnologia do Instituto Politécnico de Castelo Branco

Orientador

Professor Doutor Pedro Miguel Baptista Torres, Professor Adjunto
Escola Superior de Tecnologia do Instituto Politécnico de Castelo Branco

Agradecimentos

Aproveito esta secção para agradecer a todas as pessoas e entidades que foram parte integrante desta jornada, em especial à minha mãe, Carina Ribeiro, e aos meus avós, Vitor e Alice, por todo o apoio prestado ao longo destes 5 anos e por sempre me fazerem ver a luz ao fundo do túnel, também aos meus colegas de turma com quem uni forças para ultrapassar barreiras e, em especial, ao Manuel Filipe pela amizade e camaradagem.

Agradecer também ao Instituto Politécnico de Castelo Branco e a todos os docentes do CTeSP em Automação e Gestão Industrial e da Licenciatura em Engenharia e Gestão Industrial, cuja disponibilidade, entrega e amizade acabaram por me tornar numa pessoa com mais conhecimento e, em simultâneo, com um maior sentido de responsabilidade. Obrigado por me terem ajudado a crescer, não só como aluno, mas também como pessoa!

Um agradecimento especial ao meu orientador, Professor Doutor Pedro Torres, cujo apoio, motivação e oportunidades foram fundamentais no decorrer desta reta final e não posso deixar de agradecer às empresas CEI-Zipor e Solancis pela disponibilidade, simpatia e confiança no trabalho desenvolvido.

Este trabalho foi apoiado pela União Europeia no âmbito do *NextGeneration* EU, através de uma subvenção do Plano de Recuperação e Resiliência (PRR), no âmbito do projeto PRODUTECH R3 - “Agenda Mobilizadora da Fileira das Tecnologias de Produção para a Reindustrialização”. Código de operação: 01/C05-i11/2024.PC645808870-00000067.



Resumo

A crescente digitalização dos processos industriais tem impulsionado o desenvolvimento de sistemas capazes de monitorizar continuamente o estado dos equipamentos e antecipar possíveis falhas. Neste contexto, a Manutenção Preditiva assume um papel fundamental ao permitir a identificação precoce de comportamentos anómalos, reduzindo paragens não planeadas e aumentando a eficiência operacional dos equipamentos.

O presente trabalho apresenta o desenvolvimento de uma abordagem baseada em *Data Analytics*, *Machine Learning* e *Deep Learning* para a deteção de anomalias em sinais de vibração provenientes de um equipamento industrial de corte de pedra ornamental. A metodologia implementada inclui a aquisição e tratamento dos dados, implementação do semáforo de acordo com a ISO/TR 17243-1, preparação das séries temporais e aplicação de diferentes modelos de deteção de anomalias, com especial foco em arquiteturas de aprendizagem não supervisionada. Foram analisados modelos baseados em métodos estatísticos, algoritmos tradicionais de Machine Learning e redes neuronais profundas, incluindo *Autoencoders*, *Variational Autoencoders*, modelos convolucionais e arquiteturas orientadas para análise temporal.

A avaliação dos modelos permitiu comparar diferentes estratégias de deteção através de métricas de desempenho, não só de deteção como de eficiência computacional, identificando as vantagens e limitações de cada abordagem para este tipo de aplicação. Adicionalmente, foi considerada a implementação numa arquitetura Edge Computing, permitindo aproximar o processamento dos dados da fonte de aquisição e possibilitando uma resposta mais rápida em ambiente industrial.

Os resultados obtidos demonstram o potencial das técnicas de inteligência artificial aplicadas à Manutenção Preditiva, permitindo transformar dados de vibração em informação útil para apoio à decisão e contribuindo para a criação de sistemas industriais mais eficientes, autónomos e robustos.

Palavras-chave

Manutenção Preditiva;

Machine Learning;

Edge Computing;

Indústria 4.0;

Deteção de anomalias.

Abstract

The increasing digitalization of industrial processes has driven the development of systems capable of continuously monitoring equipment condition and anticipating potential failures. In this context, Predictive Maintenance plays a fundamental role by enabling the early identification of abnormal behaviours, reducing unplanned downtime, and improving equipment operational efficiency.

This work presents the development of an approach based on Data Analytics, Machine Learning, and Deep Learning for anomaly detection in vibration signals collected from an industrial ornamental stone cutting machine. The implemented methodology includes data acquisition and processing, the implementation of a traffic light system according to ISO/TR 17243-1, time-series preparation, and the application of different anomaly detection models, with particular focus on unsupervised learning architectures. Models based on statistical methods, traditional Machine Learning algorithms, and deep neural networks were analysed, including Autoencoders, Variational Autoencoders, convolutional models, and architectures focused on temporal analysis.

The evaluation of the models enabled the comparison of different anomaly detection strategies through performance metrics, not only regarding detection capability but also computational efficiency, identifying the advantages and limitations of each approach for this type of application. Additionally, the implementation within an Edge Computing architecture was considered, enabling data processing closer to the acquisition source and allowing faster responses in an industrial environment.

The results obtained demonstrate the potential of artificial intelligence techniques applied to Predictive Maintenance, enabling the transformation of vibration data into useful information for decision support and contributing to the development of more efficient, autonomous, and robust industrial systems

Keywords

Predictive Maintenance;

Machine Learning;

Edge Computing;

Industry 4.0;

Anomaly detection.

Índice geral

1. Introdução	1
2. Revisão da Literatura e Estado da Arte.....	5
2.1 Detecção de Anomalias em Sistemas Industriais	5
2.2 Métodos tradicionais de deteção de falhas	5
2.3 Métodos de Machine Learning para deteção de anomalias.....	6
2.4 Abordagens baseadas em Deep Learning	7
2.5 Modelos baseados em Redes Recorrentes	8
2.6 Modelos baseados em Transformers.....	8
2.7 Abordagens híbridas e métodos recentes	9
2.8 Síntese do estado da arte.....	9
3. Metodologia.....	11
3.1 ISO/TR 17243.....	12
3.2 Modbus	15
3.2.1 Diagslave	15
3.2.2 Modpoll	17
3.2.3 Simulação	18
3.3 Machine Learning.....	19
3.3.1 COPOD.....	20
3.3.2 ECOD.....	21
3.3.3 HBOS.....	21
3.4 Deep Learning	22
3.4.1 Autoencoder.....	23
3.4.2 Variational Autoencoder (VAE)	25
3.4.3 Conv1D VAE.....	27
3.4.4 DAGMM.....	29
3.4.5 TCN-AE.....	30
3.4.6 Transformer AE	32
3.4.7 USAD.....	34
3.4.8 GRU.....	36
3.4.9 LSTM	38
3.4.10 Transformers (TranAD).....	40
4. Resultados	43
4.1 Semáforo ISO/TR 17243-1.....	43

4.2 Detecção de Anomalias.....	46
5. Conclusão	55
6. Anexos	56
Referências	57

Índice de figuras

Figura 1 – CEI StoneCut	2
Figura 2 – Sensorização do <i>Spindle</i>	3
Figura 3 – Diagrama Geral do Projeto	11
Figura 4 – Implementação do Semáforo (ISO/TR 17243-1) em Python	14
Figura 5 – <i>Diagslave</i>	16
Figura 6 – <i>Modpoll</i>	17
Figura 7 – Simulação via <i>Diagslave/Modpoll</i>	18
Figura 8 – API COPOD	20
Figura 9 – Arquitetura Autoencoder	23
Figura 10 – Modelo <i>Autoencoder</i>	24
Figura 11 – <i>Threshold</i> e Classificação de Anomalia.....	25
Figura 12 – Arquitetura Variational Autoencoder	25
Figura 13 – Modelo Variational Autoencoder	26
Figura 14 – Modelo Conv1D_VAE.....	28
Figura 15 – Arquitetura DAGMM [18]	29
Figura 16 – Arquitetura TCN-AE [19].....	30
Figura 17 – Modelo TCN-AE	31
Figura 18 – Arquitetura Transformer AE	32
Figura 19 – Modelo Transformer AE	33
Figura 20 – Arquitetura USAD [20].....	34
Figura 21 – Modelo USAD.....	35
Figura 22 – Arquitetura GRU	36
Figura 23 – Modelo GRU.....	37
Figura 24 – Arquitetura LSTM	38
Figura 25 – LSTM Gates	39
Figura 26 – Modelo LSTM	40
Figura 27 – Arquitetura TranAD.....	41
Figura 28 – Resultado da Simulação do Semáforo com Gráfico via <i>Modbus</i> . 43	
Figura 29 – Resultado da Simulação do Semáforo Implementado via <i>Modbus</i>	43
Figura 30 – Exemplificação do Semáforo Implementado	44
Figura 31 – StoneCut com Semáforo na Solancis.....	45
Figura 32 – StoneCut com Semáforo na Solancis (Detalhe)	45
Figura 33 – Dataset 1	49
Figura 34 – <i>Dataset 2</i>	50
Figura 35 – Dataset 3	51
Figura 36 – Resultados Detecção de Anomalias	54

Índice de símbolos

kW – Kilowatt

kHz – Kilohertz

m/s² – meter/second²

Lista de tabelas

Tabela 1 – ISO/TR 17243-1 - Spindles with rolling element bearings and integral drives - Evaluation zone boundary values	13
Tabela 2 – Zonas de Funcionamento ISO/TR 17243-1	14
Tabela 3 – Resultados das Métricas de Avaliação Dataset 1	49
Tabela 4 – Resultados das Métricas de Avaliação Dataset 2	51
Tabela 5 – Resultados das Métricas de Avaliação Dataset 3	52

Lista de abreviaturas, siglas e acrónimos

AE – *Autoencoder*

API – *Application Programming Interface*

COPOD – *Copula-based Outlier Detection*

DAGMM – *Deep Autoencoding Gaussian Mixture Model*

ECDF – *Empirical Cumulative Distribution Functions*

ECOD – *Empirical Cumulative Outlier Detection*

EDA – *Exploratory Data Analysis*

GAN – *Generative Adversarial Network*

GMM – *Gaussian Mixture Model*

GPU – *Graphics Processing Unit*

GRU – *Gated Recurrent Unit*

HBOS – *Histogram-based Outlier Detection*

IF – *Isolation Forest*

ISO / TR – *International Organization for Standardization / Technical Reports*

LOF – *Local Outlier Factor*

LSTM – *Long-Short-Term Memory*

LTSC – *Long-Term Spindle Condition*

MSE – *Mean Squared Error*

PCAT – *Point-Correlate Adversarial Transformer*

PyOD – *Python Outlier Detection*

RBF – *Radial Basis Function*

ReLU – *Rectified Linear Unit*

RMS – *Root Mean Square*

RNN – *Recurrent Neural Networks*

RPM – *Rotation Per Minute*

SNN – *Spiking Neural Network*

STSC – *Short-Term Spindle Condition*

SVM – *Support Vector Machines*

TCN – *Temporal Convolutional Network*

TCP / IP – *Transmission Control Protocol / Internet Protocol*

TPU – *Tensor Processing Unit*

USAD – *UnSupervised Anomaly Detection*

VAE – *Variational Autoencoder*

1. Introdução

A manutenção dos ativos industriais representa um dos principais desafios nos sistemas produtivos modernos, uma vez que a disponibilidade e fiabilidade dos equipamentos influenciam diretamente a eficiência operacional, a qualidade da produção e os custos associados a paragens não planeadas. As estratégias tradicionais de manutenção preventiva baseiam-se frequentemente em planos de intervenção definidos por períodos de funcionamento ou número de ciclos dos equipamentos, não considerando necessariamente o estado real dos mesmos no momento da intervenção.

Neste contexto, a manutenção baseada na condição¹ (*Condition-Based Maintenance*) surge como uma abordagem orientada para o acompanhamento contínuo do estado dos equipamentos, permitindo ajustar as intervenções às condições reais de funcionamento. Através da aquisição e análise de variáveis físicas relevantes, como vibração, temperatura ou corrente elétrica, torna-se possível identificar alterações no comportamento normal dos sistemas e apoiar decisões de manutenção mais informadas.

A implementação deste tipo de sistemas apresenta, contudo, desafios relacionados com a aquisição, processamento e interpretação dos dados recolhidos. Em ambientes industriais, a obtenção de grandes volumes de dados representativos de falhas devidamente catalogadas é uma tarefa complexa, uma vez que as avarias correspondem normalmente a eventos pouco frequentes. Esta limitação dificulta a aplicação de metodologias de aprendizagem supervisionada, que dependem da existência de exemplos previamente classificados para diferentes modos de falha.

Neste cenário, as técnicas de deteção de anomalias baseadas em aprendizagem não supervisionada apresentam-se como uma alternativa adequada, permitindo caracterizar o comportamento normal do equipamento através dos dados históricos disponíveis e identificar desvios relativamente a esse padrão. A utilização de algoritmos de *Machine Learning* e *Deep Learning* possibilita a extração automática de características relevantes dos sinais monitorizados, contribuindo para o desenvolvimento de sistemas de apoio à manutenção mais adaptáveis e eficientes.

Esta abordagem assume particular relevância em equipamentos industriais rotativos, como motores e *spindles*, onde os sinais de vibração constituem uma fonte de informação essencial para avaliar o estado de funcionamento. Alterações nos padrões vibratórios podem estar associadas a fenómenos como desgaste mecânico, desequilíbrios, desalinhamentos ou degradação progressiva de componentes. Desta forma, a análise contínua destes sinais permite acompanhar a condição do equipamento sem necessidade de interromper o processo produtivo.

¹ <https://www.ibm.com/think/topics/condition-based-maintenance>, consultado em Junho de 2026

No entanto, a transformação dos dados recolhidos pelos sensores em informação útil para manutenção requer capacidade de processamento adequada. Os sinais industriais apresentam frequentemente características complexas, incluindo ruído, variações associadas às condições de operação e padrões temporais difíceis de identificar através de métodos convencionais. Por este motivo, a utilização de técnicas de análise de dados e inteligência artificial tem ganho relevância, permitindo complementar métodos tradicionais baseados em limiares fixos com modelos capazes de se adaptar ao comportamento real dos equipamentos.

A integração destas tecnologias em ambientes industriais introduz ainda um desafio adicional relacionado com a arquitetura de processamento dos dados. Embora as soluções baseadas em *Cloud Computing* apresentem vantagens ao nível da capacidade computacional e de armazenamento, a transmissão contínua de grandes volumes de informação pode introduzir atrasos e dependência da infraestrutura de comunicação. Em aplicações industriais onde é necessária uma resposta rápida, a utilização de *Edge Computing* permite executar o processamento e a análise junto da fonte de aquisição, reduzindo a latência e possibilitando uma monitorização mais eficiente. Esta abordagem enquadra-se nos princípios da Indústria 4.0, onde os equipamentos industriais passam a integrar capacidades de perceção, análise e decisão.

É neste enquadramento que surge o presente trabalho, desenvolvido no âmbito do Work Package 2 (WP2) – *Design.2Transform*, integrado no projeto PRODUTECH R3, cujo principal objetivo consiste na promoção da transformação digital da indústria através da adoção de tecnologias avançadas e soluções inteligentes. Neste contexto, o trabalho incidiu sobre o conceito de *Design for Edge Services*, explorando a implementação de serviços de monitorização e análise diretamente ao nível dos dispositivos de *edge computing*.

A Figura 1 ilustra o equipamento de corte de pedra desenvolvido pela CEI – Zipor, instalado na Solancis, de onde foram recolhidos os dados de vibração e onde foi aplicado o semáforo.



Figura 1 – CEI StoneCut ²

² <https://www.ceigroup.net/stone>, consultado em Junho de 2026

A concretização desta abordagem materializou-se na implementação, no controlador de uma máquina de corte de pedra da CEI (*StoneCut*), de um sistema de monitorização em tempo real da severidade das vibrações do motor *spindle*, baseado num indicador visual do tipo semáforo e fundamentado nos critérios definidos pela norma ISO/TR 17243-1. O spindle foi equipado com sensores de vibração e de temperatura para monitorização do estado do mesmo.

A Figura 2 ilustra o *spindle* com a respetiva localização dos sensores.

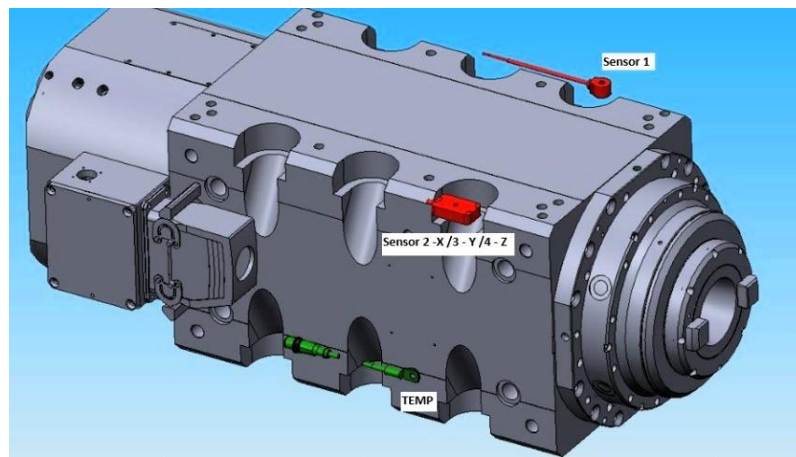


Figura 2 – Sensorização do *Spindle* ³

Paralelamente, foram realizados o desenvolvimento e a avaliação de treze algoritmos de *Machine Learning* para deteção de anomalias, recorrendo a dados de vibração adquiridos durante o funcionamento da máquina. Foram analisados indicadores de desempenho, tempos de treino e inferência, bem como os requisitos computacionais de cada modelo, com o objetivo de avaliar a sua adequação e viabilidade de implementação em dispositivos *edge*. Os resultados obtidos contribuíram para a validação de estratégias de monitorização e manutenção preditiva em ambiente industrial, aliados aos princípios da Indústria 4.0 e da computação na periferia da rede. A implementação e validação da solução desenvolvida decorreram em ambiente industrial real, nas instalações da Solancis, na Benedita. Houve recolha de dados em cenários reais de operação da máquina da CEI, treino de modelos de ML com esses dados, implementação de algoritmos em *edge PC* do IPCB (*Advantech EPC-R7300*) e, posteriormente, implementação do semáforo no controlador da máquina instalada na Solancis. Os resultados obtidos foram apresentados e demonstrados no dia 18 de junho de 2026, perante os parceiros do consórcio, o perito de avaliação externo do projeto e avaliadores do IAPMEI.

³ Confidencial da CEI – Zipor

2. Revisão da Literatura e Estado da Arte

2.1 Detecção de Anomalias em Sistemas Industriais

A detecção de anomalias em sistemas industriais é um elemento fundamental nas estratégias de manutenção preditiva e na monitorização do estado dos equipamentos. Em particular, a análise de sinais de vibração é amplamente utilizada para identificar falhas em máquinas rotativas numa fase inicial, uma vez que estes sinais contêm informações relevantes sobre o estado mecânico do equipamento [1]. A crescente disponibilidade de grandes conjuntos de dados provenientes de sensores industriais também impulsionou o desenvolvimento de métodos automáticos capazes de analisar grandes volumes de dados de vibração e identificar de forma eficiente padrões anómalos [2].

Em geral, uma anomalia pode ser definida como uma observação que se desvia significativamente do comportamento esperado de um sistema [3]. Estas podem ser classificadas em três categorias: pontuais, contextuais e coletivas. As anomalias pontuais correspondem a observações individuais que se destacam claramente do restante do conjunto de dados. As anomalias contextuais dependem do contexto em que ocorrem, em que um dado valor pode ser considerado normal ou anómalo dependendo da situação. As anomalias coletivas, por outro lado, correspondem a um conjunto de observações que, embora pareçam normais individualmente, exibem comportamentos anómalos quando analisadas em conjunto [3].

Nos sistemas industriais modernos, os dados de sensores são frequentemente representados como séries temporais multivariadas. Neste contexto, a detecção de anomalias torna-se um problema particularmente desafiante devido à elevada dimensionalidade dos dados, às dependências temporais complexas e à presença de ruído nos sinais adquiridos [4].

2.2 Métodos tradicionais de detecção de falhas

Com base nos desafios acima descritos, é importante considerar como a detecção de falhas tem sido tradicionalmente implementada em máquinas industriais, principalmente por meio de técnicas de processamento de sinais e de análise estatística. Entre as abordagens mais amplamente utilizadas estão as análises nos domínios do tempo e da frequência.

No domínio do tempo, vários indicadores estatísticos são frequentemente utilizados para caracterizar o comportamento dos sinais de vibração, incluindo a **Raiz do Quadrado Médio (RMS)**, variância, Fator de Crista, Curtose e Assimetria [5]. Estes indicadores permitem identificar alterações no comportamento dos sinais que podem indicar defeitos mecânicos.

No domínio da frequência, métodos como a análise do envelope⁴ baseada na Transformada de Hilbert têm-se revelado altamente eficazes para detetar falhas em rolamentos. Este método permite a extração de excitações periódicas presentes em sinais de vibração associados a defeitos mecânicos específicos [5].

Apesar da sua utilidade, estas abordagens apresentam algumas limitações importantes. Em particular, frequentemente requerem um elevado nível de conhecimento especializado para selecionar e interpretar as características relevantes do sinal. Além disso, a sua capacidade de generalização é limitada quando aplicada a sistemas complexos ou quando os padrões de falha são desconhecidos.

2.3 Métodos de Machine Learning para deteção de anomalias

Com base em abordagens tradicionais, avanços na capacidade computacional e na disponibilidade de dados industriais impulsionaram o desenvolvimento de métodos de Machine Learning para a deteção automática de anomalias.

Entre os métodos tradicionais de *Machine Learning*, destacam-se algoritmos como *Support Vector Machines (SVM)*, *Isolation Forest (IF)* e *Local Outlier Factor (LOF)*. Estes algoritmos identificam padrões anómalos por meio da análise da distribuição dos dados ou da construção de limites de decisão que separam dados normais de anómalos [6].

O algoritmo *Isolation Forest* identifica anomalias construindo árvores de decisão que isolam rapidamente observações raras no conjunto de dados. O método do *Local Outlier Factor*, por sua vez, baseia-se na densidade local dos vizinhos para identificar observações com densidade significativamente diferente da dos seus vizinhos [6].

Outra abordagem frequentemente utilizada é a aplicação de *One-Class Support Vector Machines (One-Class SVM)*, que aprendem um limite que abrange o comportamento normal do sistema e classificam observações fora desta região como anómalas [1].

Apesar de apresentarem bons resultados em certos cenários, estes métodos têm dificuldades quando aplicados a séries temporais complexas, pois nem sempre conseguem captar dependências temporais de longo alcance presentes nos dados.

⁴ A análise do envelope, ou *Envelope Analysis*, é a extração da envolvente de um sinal de vibração, ou seja, da curva que representa a variação de amplitude de um sinal de vibração ao longo do tempo, para realçar impactos periódicos provocados por defeitos em componentes rotativos.

2.4 Abordagens baseadas em Deep Learning

Os modelos de *Deep Learning* têm demonstrado recentemente resultados promissores na detecção de anomalias em séries temporais, especialmente em aplicações industriais. Estudos comparativos indicam que as arquiteturas de *Deep Learning*, incluindo *Autoencoders*, redes recorrentes e modelos baseados em mecanismos de atenção, podem apresentar desempenhos distintos dependendo do tipo de dados e da aplicação considerada [7].

Uma das abordagens mais amplamente utilizadas envolve o uso de *Autoencoders*, que são redes neurais treinadas para reconstruir dados de entrada a partir de uma representação comprimida. Neste contexto, assume-se que o modelo aprende a reconstruir corretamente apenas os padrões correspondentes ao comportamento normal do sistema. Assim, quando confrontado com dados anômalos, o erro de reconstrução tende a aumentar significativamente, o que permite identificar anomalias [8].

Várias variantes desta arquitetura foram propostas na literatura. Por exemplo, em [5], é apresentado um *Robust Deep Autoencoder* capaz de extrair características não lineares relevantes mesmo quando os dados de treino contêm ruído. Neste trabalho, as funcionalidades extraídas pelo *Autoencoder* são posteriormente utilizadas por um classificador baseado em *Support Vector Machine* para detecção de falhas.

Outra abordagem utiliza *Convolutional Autoencoders*, que empregam operações convolucionais para capturar padrões locais nos sinais. Estas arquiteturas são particularmente eficazes quando aplicadas a representações em tempo-frequência de sinais, como espectrogramas ou escalogramas [8].

Além disso, alguns estudos exploram o uso de *Convolutional Autoencoders* unidimensionais aplicados diretamente a séries temporais de vibração, permitindo capturar relações entre diferentes pontos da sequência sem necessidade de engenharia manual de características [9].

Outras abordagens baseadas em reconstrução utilizam modelos probabilísticos mais complexos, como os *Deep State Space Models*, capazes de modelar explicitamente a dinâmica temporal dos sistemas e de melhorar a detecção de anomalias em sinais de vibração [10].

2.5 Modelos baseados em Redes Recorrentes

As *Recurrent Neural Networks (RNN)* e as suas variantes têm sido amplamente utilizadas para modelar dependências temporais em séries temporais.

Entre estas arquiteturas, as redes *Long-Short-Term Memory (LSTM)* destacam-se pela capacidade de capturar dependências de longo prazo, mitigando problemas como o problema do gradiente de nulidade observado nas *RNNs* tradicionais. Em tarefas de detecção de anomalias, estes modelos podem ser usados para prever valores futuros de uma série temporal, sendo o erro de previsão um indicador de anomalias [11].

No entanto, abordagens baseadas apenas em *LSTM* podem ter dificuldades em captar simultaneamente dependências temporais em diferentes escalas. Para ultrapassar esta limitação, [11] propõe um modelo *Dynamic-Input LSTM (DI-LSTM)* capaz de lidar com múltiplos sinais de entrada e de extrair características temporais mais complexas. Resultados experimentais demonstram que este modelo alcança desempenho superior em comparação com arquiteturas baseadas em *LSTMs* tradicionais e em *Autoencoders LSTM*.

2.6 Modelos baseados em Transformers

Mais recentemente, arquiteturas baseadas em *Transformers* têm sido exploradas para a detecção de anomalias em séries temporais.

Os *Transformers* foram originalmente propostos para tarefas de processamento de linguagem natural, mas a sua capacidade de modelar dependências de longo alcance por meio de mecanismos de autoatenção torna-os particularmente adequados para a análise de séries temporais [3].

Entre os modelos mais relevantes, destaca-se o *Anomaly Transformer*, que introduz o conceito de discrepância de associação para distinguir padrões normais de anómalos em séries temporais. Este modelo utiliza mecanismos de atenção para captar relações entre diferentes pontos da sequência, permitindo a identificação de padrões anómalos com alta precisão [3].

Foram propostas várias extensões desta arquitetura. Por exemplo, em [4], é apresentado o modelo *RESTAD*, que combina o erro de reconstrução com um mecanismo de similaridade baseado em funções de base radial (*RBF*), melhorando a detecção de anomalias subtis que poderiam ser corretamente reconstruídas por modelos tradicionais.

Outra abordagem promissora é o *Point-Correlate Adversarial Transformer (PCAT)*, que combina mecanismos de atenção com treino adversarial para captar correlações complexas em séries temporais multivariadas, alcançando resultados superiores em comparação com vários modelos de referência [12].

2.7 Abordagens híbridas e métodos recentes

Para melhorar a robustez e a capacidade de generalização dos modelos, vários trabalhos recentes exploram arquiteturas híbridas que combinam diferentes técnicas de aprendizagem. Por exemplo, algumas soluções integram *Deep Autoencoders* com classificadores tradicionais, como as *Support Vector Machines*, aproveitando a extração automática de características para melhorar a separação entre padrões normais e anômalos [15]. Outras abordagens combinam transformadas tempo-frequência, como a Transformada Contínua de Wavelet, com redes generativas adversárias profundas (GANs), permitindo a detecção de anomalias nas representações tempo-frequência de sinais de vibração [13]. Além disso, modelos como o *TransGANomaly* integram *Transformers* com redes GAN, beneficiando-se tanto da modelação de dependência temporal quanto da capacidade de geração adversarial para maior robustez na presença de ruído [14]. Estudos indicam que estas arquiteturas híbridas tendem a alcançar ganhos de precisão e adaptabilidade, particularmente em ambientes industriais com dados complexos e ruidosos.

Além disso, alguns trabalhos exploram arquiteturas neuromórficas e implementações eficientes em hardware para a detecção de anomalias em tempo real. Um exemplo é o *TinySNN*, uma rede neural baseada em neurónios de pico, implementada em hardware, para detecção não supervisionada de anomalias com baixo consumo energético [16].

2.8 Síntese do estado da arte

A revisão da literatura demonstra que a detecção de anomalias em séries temporais industriais continua a ser um problema de investigação ativo. Embora os métodos tradicionais baseados em análise de sinais e em algoritmos clássicos de *Machine Learning* continuem a ser utilizados, as abordagens baseadas em *Deep Learning* têm demonstrado desempenho superior na modelação de padrões complexos presentes nos dados. No entanto, vários desafios e lacunas na investigação permanecem por resolver. Entre as principais questões em aberto estão: a dificuldade em detetar anomalias subtis em sinais muito ruidosos; a capacidade limitada dos modelos atuais de generalizar a diferentes tipos de equipamentos ou cenários operacionais; a elevada dependência de grandes volumes de dados rotulados ou dados representativos do comportamento normal; a falta de interpretabilidade de modelos avançados, como *Transformers* e redes adversariais; e a escassez de soluções eficientes para implementação em tempo real e em hardware com recursos limitados. Estas questões representam oportunidades para desenvolvimentos futuros e motivam a continuidade da investigação na área.

Entre estas abordagens, os *Autoencoders* continuam a ser amplamente utilizados devido à sua simplicidade e eficácia em cenários em que apenas dados

normais estão disponíveis. As principais vantagens dos *Autoencoders* incluem a facilidade de implementação, baixo custo computacional e a capacidade de detetar anomalias sem exigir grandes volumes de dados rotulados, o que os torna especialmente úteis quando o objetivo é captar padrões gerais de normalidade e quando os recursos computacionais são limitados. No entanto, os *Autoencoders* podem ter dificuldades em captar relações temporais de longo alcance ou padrões globais em séries temporais multivariadas e também são sensíveis ao ruído nos dados.

Por outro lado, as arquiteturas baseadas em *Transformers* têm vindo a ganhar destaque devido à sua capacidade de modelar dependências temporais complexas e de captar relações globais em séries temporais. Estas redes beneficiam de mecanismos de autoatenção, permitindo a identificação de padrões subtis e de interações entre diferentes variáveis e escalas temporais. Apesar disso, os *Transformers* têm um custo computacional mais elevado, geralmente requerem mais dados para um treino eficaz e a interpretação dos resultados pode ser mais desafiadora. No contexto industrial, a escolha entre *Autoencoders* e *Transformers* depende do volume de dados disponíveis, da complexidade dos padrões de anomalia a detetar e das restrições computacionais relativas aos recursos disponíveis.

Apesar dos avanços alcançados, persistem desafios importantes, incluindo a deteção de anomalias subtis, a robustez ao ruído e a interpretação dos resultados obtidos pelos modelos. Estas limitações continuam a motivar o desenvolvimento de novas abordagens capazes de melhorar a precisão e fiabilidade dos sistemas de deteção de anomalias em ambientes industriais.

3. Metodologia



Figura 3 – Diagrama Geral do Projeto

A Figura 3 representa o diagrama geral do projeto com as principais fases, desde a medição e aquisição de dados de vibração até à representação dos dados no formato de semáforo e à aplicação dos dados de vibração em modelos de *Machine Learning*.

Numa fase inicial de testes, os dados foram recolhidos no equipamento a uma frequência de amostragem de 50 kHz através do *software* IFM VES004, exportados para ficheiro .csv e tratados em Python, onde foi aplicado o semáforo de acordo com a *ISO/TR 17243-1*. Como saída, foi gerado um gráfico com os valores de vibração obtidos, distribuídos no tempo e assinalados com a respetiva cor do semáforo.

Na fase de implementação, os dados são enviados do equipamento para o *Edge computer* através de *Modbus*, onde é executado o ficheiro *Python* com os mesmos critérios do semáforo e são apresentados na *HMI* do equipamento com *timestamp*, valor em m/s^2 , estado e cor de fundo, de acordo com o mesmo, para facilitar a rápida interpretação por parte do operador e, por baixo, o histórico dos seis últimos valores recebidos. A informação é atualizada de dois em dois segundos.

Para a parte de *Machine Learning*, foram utilizados os ficheiros .csv como entrada, divididos os dados em janelas de 1024 valores devido à dependência temporal e sequencial de um sinal de vibração, normalizados os dados conforme o algoritmo aplicado, executado o algoritmo, gerado um gráfico de deteção de anomalias, métricas de avaliação da eficiência computacional do algoritmo e indicadores de desempenho.

3.1 ISO/TR 17243

As normas ISO são acordadas internacionalmente por especialistas. São vistas como uma fórmula que descreve a melhor forma de fazer algo. Pode tratar-se de criar um produto, gerir um processo, prestar um serviço ou fornecer materiais – as normas abrangem uma vasta gama de atividades. Os padrões são a sabedoria destilada de pessoas com experiência na sua área e que conhecem as necessidades das organizações que representam – como fabricantes, vendedores, compradores, clientes, associações comerciais, utilizadores ou reguladores.⁵

A norma ISO/TR 17243-1 fornece informações sobre como avaliar a gravidade das vibrações do *spindle*, medidas na carcaça do mesmo. Os critérios fornecidos aplicam-se a *spindles* com acionamento integrado, utilizados em máquinas-ferramenta estacionárias, com velocidades nominais de funcionamento entre 6000 e 30000 RPM. Esta parte da ISO/TR 17243 aplica-se apenas a *spindles* com tipos de rolamentos de elementos rolantes.⁶

Após contacto com o fornecedor do *spindle*, foi possível confirmar que o equipamento em questão está equipado com dois pares de rolamentos de esferas: um par de 7020 à frente e outro de 7014 atrás. Apesar do fabricante dos rolamentos não ter sido especificado, ambos apresentam elevada capacidade de carga e rigidez e são aplicáveis a casos de velocidade média.^{7 8}

Por meio do *datasheet*⁹ do *spindle* VEM MTB24 6A, foi possível obter informações relevantes para a aplicação correta da norma, como a potência e a velocidade de rotação máxima.

Sendo o *spindle* em questão de 26 kW com uma velocidade máxima de 8000 rpm, acionamento direto e rolamentos de esferas, a coluna a consultar deve ser “Rated power > 5 kW ball bearing (point contact)”, pertencente à **Tabela 3** da ISO/TR 17243-1.

⁵ <https://www.iso.org/standards.html>, consultado em Março de 2026

⁶ <https://www.iso.org/standard/59471.html>, consultado em Março de 2026

⁷ <https://www.abf.store/s/en/bearings/B7020-C-T-P4S-UL-FAG/350244>, consultado em Março de 2026

⁸ <https://www.abf.store/s/en/bearings/B7014-C-T-P4S-UL-FAG/350234>, consultado em Março de 2026

⁹ <https://vem.it/wp-content/uploads/MTB-24-rev1.pdf>, consultado em Março de 2026

Tabela 1 – ISO/TR 17243-1 - Spindles with rolling element bearings and integral drives - Evaluation zone boundary values ¹⁰

Zone boundary	Unit	Spindle classification			
		Rated power ≤ 5 kW ball bearing (point contact)	Rated power > 5 kW ball bearing (point contact)	Rated power ≤ 5 kW roller bearing (line contact)	Rated power > 5 kW roller bearing (line contact)
a	mm/s RMS	0,7	0,7	0,7	0,7
b	mm/s RMS	1,1	1,4	1,1	1,4
c	mm/s RMS	1,8	2,8	1,8	2,8
i	m/s ² RMS	6	6	8	8
ii	m/s ² RMS	10	10	13	13
iii	m/s ² RMS	15	15	20	20
iv	m/s ² RMS	20	20	26	26
v	m/s ² RMS	25	25	32	32
vi	m/s ² RMS	30	30	39	39
vii	m/s ² RMS	40	40	52	52
viii	m/s ² RMS	50	50	65	65

Os valores da Tabela 1 vão determinar os limites das zonas de avaliação para a velocidade do *spindle* em questão na Tabela 2 da ISO/TR 17243-1.

Para uma velocidade nominal de rotação de 8000 rpm, a coluna a consultar é a 2ª coluna da Tabela 2, que compreende o intervalo “6 000 < r/min ≤ 12 000”.

Para determinar o parâmetro STSC, a medição é realizada em aceleração (m/s²) no intervalo de 2 kHz a 10 kHz.

Desta forma, segundo a ISO/TR 17243-1, a zona A compreende valores até 10 m/s² RMS, é representada na cor verde e é associada a eixos recentemente colocados em serviço. A zona B compreende valores entre 10 m/s² RMS e 20 m/s² RMS, sendo representada em amarelo e associada a condições normais de funcionamento. A zona C compreende valores entre 20 m/s² RMS e 40 m/s² RMS, representada na cor laranja e associada a amplitudes de vibração elevadas, não adequada à operação a longo prazo. Por fim, a zona D refere-se a valores acima de 40 m/s² RMS, representada em vermelho e associada a amplitudes vibracionais inaceitáveis, condição crítica.

¹⁰ <https://www.vibraconseil.fr/normes/norme%20ISO%2017243.htm>, consultado em Março de 2026

Tabela 2 – Zonas de Funcionamento ISO/TR 17243-1

		600 < r/min ≤ 6 000	6 000 < r/min ≤ 12 000	12 000 < r/min ≤ 18 000	18 000 < r/min ≤ 30 000
LTSC (mm/s RMS)	a	A			
	b	B			
	c	C			
	d	D			
STSC (m/s ² RMS)	i	A	A	A	A
	ii	B	B	A	A
	iii	B	B	B	A
	iv	C	C	B	B
	v	C	C	C	B
	vi	C	C	C	C
	vii	D	D	D	C
	viii	D	D	D	D

A implementação do semáforo em Python, de acordo com as cores e os intervalos da norma, foi realizada conforme se observa na figura abaixo.

Cada zona foi atribuída à respetiva cor e a uma classificação representativa do estado da máquina, indo de “GOOD” para a zona A a “UNACCEPTABLE” para a zona D. No caso de não existirem dados de vibração provenientes da máquina, a janela fica cinzenta com a mensagem “NO DATA”. Desta forma, o operador consegue ter uma perceção rápida e simples do estado da máquina.

```
# Semáforo (ISO/TR 17243-1)
# Spindle 26 kW, rotação 6-12k rpm
# Table 2 - General Zone Boundaries
# Limites STSC m/s2 RMS
# Table 3 (Rated Power >5kW ball bearing (point contact))
def traffic_light_rms(x):
    if x is None:
        return "gray", "NO DATA"
    if x <= 10:
        return "#2ca02c", "GOOD" # green
    elif x <= 20:
        return "#ffcc00", "ACCEPTABLE" # yellow
    elif x <= 40:
        return "#ff7f0e", "UNSATISFACTORY" # orange
    else:
        return "#d62728", "UNACCEPTABLE" # red
```

Figura 4 – Implementação do Semáforo (ISO/TR 17243-1) em Python

3.2 Modbus

No desenvolvimento deste projeto, tornou-se necessário simular a comunicação entre o sistema de aquisição de dados e um dispositivo industrial. Nesse contexto, optou-se pelo uso do protocolo Modbus TCP/IP, uma vez que é o adotado no cenário real do caso de estudo.

O protocolo Modbus ¹¹ foi originalmente desenvolvido pela Modicon no final da década de 1970 e, posteriormente, adaptado para operar sobre redes Ethernet, dando origem ao Modbus TCP/IP. Esta versão permite a troca de dados através da infraestrutura de rede convencional, o que facilita bastante a integração com sistemas informáticos, como aplicações em Python, que é o caso, ou com plataformas de monitorização.

De forma geral, o funcionamento do Modbus TCP/IP baseia-se num modelo cliente-servidor. Isto significa que há um dispositivo que faz pedidos (cliente) e outro que responde (servidor). No caso deste projeto, a aplicação desenvolvida em Python assume o papel de cliente, enquanto o servidor corresponde ao dispositivo que disponibiliza os dados — neste caso, inicialmente simulado.

Os dados são organizados em registos; os mais utilizados são os *holding registers*, que permitem a leitura e a escrita de valores numéricos. Na prática, cada registo pode representar uma variável do sistema, como, por exemplo, um valor de vibração, temperatura ou velocidade. Esta estrutura, apesar de simples, revelou-se suficiente para os objetivos do projeto e é uma das razões pelas quais o *Modbus* continua a ser amplamente utilizado.

Uma das dificuldades iniciais foi a ausência de um sistema físico disponível para testes. De forma a contornar essa limitação, recorreu-se a ferramentas de simulação, nomeadamente o *diagslave* e o *modpoll*.

3.2.1 Diagslave

O *diagslave* ¹² é um simulador e ferramenta de teste que funciona por meio de comandos no terminal, tendo sido utilizado para simular uma ligação via *Modbus* TCP/IP, assumindo o papel de servidor. Na prática, esta ferramenta comporta-se como um dispositivo industrial, disponibilizando registos que podem ser lidos ou escritos através da rede. Isto permitiu criar um ambiente de teste controlado, no qual foi possível validar a comunicação sem depender de hardware real. Esta abordagem revelou-se particularmente útil numa fase inicial, em que o foco estava na implementação da lógica de comunicação e na integração com o código desenvolvido em Python.

¹¹ <https://pt.wikipedia.org/wiki/Modbus>, consultado em Abril de 2026

¹² <https://www.modbusdriver.com/diagslave.html>, consultado em Abril de 2026

```

Linha de comandos - diagslav X + v
Microsoft Windows [Version 10.0.26200.8039]
(c) Microsoft Corporation. Todos os direitos reservados.

C:\Users\Rodrigo Ribeiro>cd Desktop\diagslave\win

C:\Users\Rodrigo Ribeiro\Desktop\diagslave\win>diagslave -h
diagslave 3.5 - FieldTalk(tm) Modbus(R) Diagnostic Slave Simulator
Copyright (c) 2002-2024 proconX Pty Ltd
Visit https://www.modbusdriver.com for Modbus libraries and tools.

Usage: diagslave [OPTIONS] [SERIALPORT]
Arguments:
SERIALPORT    Serial port when using Modbus ASCII or Modbus RTU protocol
               COM1, COM2 ...          on Windows
               /dev/ttyS0, /dev/ttyS1 ... on Linux

General options:
-m ascii      Modbus ASCII protocol
-m rtu        Modbus RTU protocol (default if SERIALPORT set)
-m tcp        MODBUS/TCP protocol (default otherwise)
-m udp        MODBUS UDP
-m enc        Encapsulated Modbus RTU over TCP
-o #          Master activity time-out in seconds (1.0 - 100, 3 s is default)
-a #          Slave address (1-255 for RTU/ASCII, 0-255 for TCP)
Options for MODBUS/TCP, UDP and RTU over TCP:
-p #          TCP port number (502 is default)
-c #          Connection time-out in seconds (1.0 - 3600, 60 s is default)
Options for Modbus ASCII and Modbus RTU:
-b #          Baudrate (e.g. 9600, 19200, ...) (19200 is default)
-d #          Databits (7 or 8 for ASCII protocol, 8 for RTU)
-s #          Stopbits (1 or 2, 1 is default)
-p none       No parity
-p even       Even parity (default)
-p odd        Odd parity
-4 #          RS-485 mode, RTS on while transmitting and another # ms after
Options for Modbus RTU:
-f #          Tolerance for inter-frame gap detection in ms (0-20)

C:\Users\Rodrigo Ribeiro\Desktop\diagslave\win>diagslave -m tcp -p 502
diagslave 3.5 - FieldTalk(tm) Modbus(R) Diagnostic Slave Simulator
Copyright (c) 2002-2024 proconX Pty Ltd
Visit https://www.modbusdriver.com for Modbus libraries and tools.

Protocol configuration: MODBUS/TCP
Slave configuration: address = -1, master activity t/o = 3.00s
IP configuration: port = 502, connection t/o = 60.00s

Server started up successfully.
Listening to network (Ctrl-C to stop)
.....|

```

Figura 5 – Diagslave

3.2.2 Modpoll

Por outro lado, o *modpoll*¹³ foi utilizado como ferramenta de apoio para testar a comunicação. Funcionando como cliente *Modbus*, permitiu enviar pedidos ao *diagslave* e verificar se os dados eram lidos e escritos corretamente. Em termos práticos, foi uma ferramenta importante para validar endereços de registo, testar diferentes valores e garantir que a comunicação funcionava como esperado antes de avançar para fases mais complexas.

```

C:\Users\Rodrigo Ribeiro\Desktop\modpoll\win
Microsoft Windows [Version 10.0.26200.8093]
(c) Microsoft Corporation. Todos os direitos reservados.

C:\Users\Rodrigo Ribeiro\Desktop\modpoll\win
C:\Users\Rodrigo Ribeiro\Desktop\modpoll\win>modpoll -h
modpoll 3.16 - FieldTalk(tm) Modbus(R) Master Simulator
Copyright (c) 2002-2025 proconX Pty Ltd
Visit https://www.modbusdriver.com for Modbus Libraries and tools.

Usage: modpoll [OPTIONS] SERIALPORT[HOST [WRITEVALUES...]]

Arguments:
SERIALPORT  Serial port when using Modbus ASCII or Modbus RTU protocol
             COM1, COM2, ... on Windows
             /dev/ttyS0, /dev/ttyS1, ... on Linux
HOST        Host name or dotted IP address when using MODBUS/TCP protocol
WRITEVALUES List of values to be written. If none specified (default) modpoll reads data.

General options:
-m ascii    Modbus ASCII protocol
-m rtu      Modbus RTU protocol (default if SERIALPORT contains \ or COM)
-t tcp      MODBUS/TCP protocol (default otherwise)
-m udp      MODBUS UDP protocol
-w enc      Encapsulated Modbus RTU over TCP
-a #        Slave address (1-247 for serial, 0-255 for TCP, 1 is default)
-r #        Start reference (0-65535, 1 is default), use -R for zero-based references (0-65535, 0 is default)
-c #        Number of values to read (1-125, 1 is default), optional force writing (use -c 1 to force FC5 or FC6)
-t #        Discrete output (coil) data type (FC 1)
-r #        Discrete input data type (FC 2)
-t #        16-bit input register data type (FC 4)
-r #        16-bit input register data type with hex display
-t #i32     32-bit integer data type in input register table
-t #i64     64-bit integer data type in input register table
-t #i3mod   32-bit module 100000 data type in input register table
-t #f32     32-bit float data type in input register table
-t #d64     64-bit double data type in input register table
-t #d4      16-bit holding register data type (FC3, default)
-t #i32hex 16-bit holding register data type with hex display
-t #i32     32-bit integer data type in holding register table
-t #i64     64-bit integer data type in holding register table
-t #i3mod   32-bit module 100000 data type in holding register table
-t #f32     32-bit float data type in holding register table
-t #d64     64-bit double data type in holding register table
-t id       Read device identification objects (FC 43/44)
-t file     File record reference type 6 (FC 20/21)
-r #        File number for file record (default is 4)
-u          Display data as unsigned integers
-i          Slave operates on big-endian 32-bit/64-bit integers
-f          Slave operates on big-endian 32-bit/64-bit floats
-x          Use Daniel/Enron single register 32-bit mode (implies -i and -f)
-x          Use Lufkin ELAM extensions (RTU and RTU over TCP only)
-R          First reference is 0 (POU addressing) instead 1

C:\Users\Rodrigo Ribeiro\Desktop\modpoll\win>

```

Figura 6 – Modpoll

¹³ <https://www.modbusdriver.com/modpoll.html>, consultado em Abril de 2026

3.2.3 Simulação

A utilização combinada destas ferramentas permitiu reproduzir, de forma bastante próxima, o comportamento de um sistema real. Em vez de depender diretamente de sensores físicos, foi possível simular valores e observar como a aplicação desenvolvida os tratava. Isto incluiu, por exemplo, a leitura de dados em tempo real, a sua representação gráfica e a aplicação de critérios de avaliação (como o sistema de “semáforo” da norma ISO/TR-17243-1 implementado).

Esta abordagem trouxe algumas vantagens claras. Desde logo, permitiu desenvolver e testar o sistema de forma independente, sem necessidade de acesso contínuo ao equipamento industrial do caso de estudo. Além disso, facilitou a identificação de erros numa fase precoce, o que reduziu o tempo de desenvolvimento global e, por fim, ao utilizar o protocolo Modbus TCP/IP, garantiu-se que a solução poderia ser posteriormente aplicada no contexto real com alterações mínimas.

Em síntese, a utilização do protocolo de comunicação Modbus TCP/IP, aliada a ferramentas de simulação como o *diagslave* e o *modpoll*, constituiu uma solução prática e eficaz para o desenvolvimento deste projeto. Para além de permitir a validação da comunicação, esta abordagem contribuiu para uma implementação mais estruturada e preparada para integração em ambiente industrial.

The image shows two terminal windows side-by-side. The left window, titled 'Linha de comandos - modpo', displays a series of commands and their outputs. The commands are: 'Polling slave... (Ctrl-C to stop)', '[1]: 0', and then the same sequence repeated 15 times. The right window, titled 'Linha de comandos - diagslav', shows a list of slave processes. Each line in this window consists of a 'Slave' label, a line number (from 1 to 15), and the text 'readHoldingRegisters from 1, 1 references'.

Figura 7 – Simulação via *Diagslave/Modpoll*

3.3 Machine Learning

Machine Learning é um subdomínio de inteligência artificial que permite que as máquinas aprendam com dados sem serem explicitamente programadas.¹⁴

É uma técnica que descobre relações previamente desconhecidas em dados por meio de algoritmos sofisticados, explorando conjuntos de dados em busca de padrões ou tendências que vão para além da análise estatística simples.¹⁵

No Machine Learning, existem 4 principais tipos de aprendizagem: Supervised, Unsupervised, Semi-supervised e Reinforcement.

A aprendizagem supervisionada requer uma distinção prévia entre dados normais e anormais, por exemplo, atribuindo uma etiqueta a cada exemplo.

A aprendizagem não supervisionada não requer uma distinção entre o normal e o anormal; baseia-se na exploração do conjunto de dados e na procura de tendências e padrões.

No caso da aprendizagem semissupervisionada, são necessários rótulos nos dados para o treino inicial do algoritmo, de modo a estabelecer parâmetros gerais do projeto. De seguida, o algoritmo utiliza esses dados rotulados para tentar avaliar e identificar dados não rotulados com alta probabilidade. Este processo é iterativo e pode ser repetido, uma vez que o conjunto de dados rotulados aumenta a cada iteração do algoritmo.

A aprendizagem por reforço funciona de forma semelhante à aprendizagem não supervisionada, uma vez que utiliza um conjunto de dados não rotulados, no entanto, concentra-se em aprender o melhor caminho para alcançar um objetivo específico através de feedback, tanto positivo como negativo ou até mesmo neutro, em vez de procurar por padrões ou tendências.

No âmbito deste projeto, optou-se por utilizar apenas aprendizagem não supervisionada, uma vez que não se sabe a priori se os dados representam uma anomalia ou o funcionamento normal do equipamento. Partimos do pressuposto de que a máquina não apresenta avarias, no entanto, observam-se picos de vibração em determinadas operações ou condições de funcionamento.

Os algoritmos de Machine Learning utilizados foram: COPOD¹⁶, ECOD¹⁷ e HBOS¹⁸.

¹⁴ <https://www.iso.org/artificial-intelligence/machine-learning>, consultado em Maio de 2026

¹⁵ <https://www.oracle.com/europe/artificial-intelligence/machine-learning/what-is-machine-learning/>, consultado em Maio de 2026

¹⁶ <https://github.com/yzhao062/pyod/blob/master/pyod/models/copod.py>, consultado em Maio de 2026

¹⁷ <https://github.com/yzhao062/pyod/blob/master/pyod/models/ecod.py>, consultado em Maio de 2026

¹⁸ <https://github.com/yzhao062/pyod/blob/master/pyod/models/hbos.py>, consultado em Maio de 2026

3.3.1 COPOD

COPOD é um algoritmo para detecção de anomalias, implementado no pacote de Python PyOD, com várias características que o fazem destacar-se de algoritmos competitivos. Este algoritmo é determinístico, ou seja, para o mesmo conjunto de dados, produzirá sempre exatamente o mesmo resultado, pois não utiliza inicializações aleatórias, não faz amostragem aleatória de dados e não depende de um seed aleatório. Logo, as pontuações de anomalia e os *outliers* identificados serão sempre os mesmos no mesmo conjunto de dados. Não tem hiperparâmetros, o que é importante, pois são difíceis de ajustar em tarefas de detecção de anomalias, uma vez que é raro saber se os dados são normais ou anómalos e, em condições normais, não é viável simular anomalias para obter dados de referência. É também um modelo rápido e eficiente em termos de computação, o ideal para aplicação em *Edge Computing*, e permite escalabilidade para conjuntos de dados de elevada dimensionalidade.¹⁹

Embora o COPOD seja um algoritmo baseado em cópulas, neste trabalho foi aplicada apenas uma variável de entrada (vibração). Assim, a detecção de anomalias baseia-se essencialmente na posição dos valores observados na distribuição empírica da vibração, não sendo exploradas dependências entre múltiplas variáveis.

```
# train the COPOD detector
from pyod.models.copod import COPOD
clf = COPOD()
clf.fit(X_train)

# get the prediction label and outlier scores of the training data
y_train_pred = clf.labels_ # binary labels (0: inliers, 1: outliers)
y_train_scores = clf.decision_scores_ # raw outlier scores

# get the prediction on the test data
y_test_pred = clf.predict(X_test) # outlier labels (0 or 1)
y_test_scores = clf.decision_function(X_test) # outlier scores
```

Figura 8 – API COPOD ²⁰

Na Figura 8, é possível perceber a base do COPOD. Na primeira linha, é importado o algoritmo da biblioteca PyOD, de seguida, é criada uma instância do detetor e treinado o modelo, sendo considerado praticamente *parameter-free*, ou seja, não precisa de afinação complexa.

¹⁹ <https://medium.com/data-science/fast-accurate-anomaly-detection-based-on-copulas-copod-3133ce9041fa>, consultado em Maio de 2026

²⁰ Retirado de <https://github.com/winstonli/COPOD/tree/master> em Maio de 2026

Os rótulos são obtidos com os valores de 0 ou 1, em que 0 são considerados dados normais e 1 dados anômalos, de seguida, são obtidos os *scores* onde, quanto maior for o score, mais anômalo é o ponto em questão.

3.3.2 ECOD

ECOD é um algoritmo de detecção de anomalias não supervisionado baseado na ideia de que os *outliers* são eventos raros que aparecem nas caudas das distribuições dos dados. O método estima a distribuição dos dados através de ECDFs e utiliza essa informação para calcular um score de anomalia para cada observação. Este algoritmo foi desenvolvido para resolver problemas como elevado custo computacional, necessidade de ajuste de hiperparâmetros e baixa interpretabilidade dos resultados, mantendo simultaneamente uma elevada precisão e escalabilidade.

O algoritmo requer três etapas principais, numa primeira fase é calculada a ECDF para cada variável do conjunto de dados, neste caso é apenas vibração, de seguida são estimadas as probabilidades de cauda para cada observação e por último são agregadas as probabilidades de cauda das diferentes dimensões para obter um score de anomalia global.

Segundo os autores, o ECOD apresenta vantagens como ser *parameter-free*, ser de fácil interpretação, não assumir distribuições específicas dos dados, eficiência computacional e escalabilidade para conjuntos de dados grandes e de elevada dimensionalidade. [17]

3.3.3 HBOS

O *Histogram-Based Outlier Score* é um algoritmo de detecção de anomalias não supervisionado que utiliza histogramas para identificar *outliers*. Ao contrário de outros métodos, o HBOS é conhecido pela sua eficiência e velocidade, não exigindo cálculos complexos e sendo adequado para grandes conjuntos de dados.

Este algoritmo funciona construindo histogramas para cada característica no conjunto de dados. Em seguida, o algoritmo calcula o grau de discrepância (*outlying*) para cada instância com base nos histogramas. Para cada característica, um histograma é construído, podendo os intervalos ter largura fixa (estáticos) ou serem determinados dinamicamente com base na distribuição dos dados. A estimativa de densidade é obtida através da frequência de amostras e a pontuação de *outlier* para cada instância é calculada com base na densidade dos intervalos em que ela se enquadra.

Uma densidade maior indica uma maior probabilidade de ser *outlier*.²¹

²¹ <https://www.geeksforgeeks.org/machine-learning/hbos-efficient-outlier-detection-with-python/>, consultado em Maio de 2026

3.4 Deep Learning

Deep Learning é um subconjunto de *Machine Learning* impulsionado por redes neurais multicamadas, cujo design é inspirado na estrutura do cérebro humano.

Ao contrário da lógica matemática explicitamente definida nos algoritmos de *Machine Learning*, as redes neurais artificiais dos modelos de *Deep Learning* compreendem muitas camadas interligadas de “neurónios”, cada um executando uma operação matemática.²²

Estas redes, frequentemente designadas por *deep neural networks*, são inspiradas no cérebro humano e caracterizam-se pela capacidade de extrair representações hierárquicas dos dados de forma automática, sem necessidade de definição manual de características.

O funcionamento destes modelos assenta na propagação dos dados ao longo das várias camadas do modelo de *Deep Learning*, nas quais vão aprendendo representações cada vez mais abstratas. As primeiras camadas costumam captar padrões simples no conjunto de dados, enquanto as camadas mais profundas conseguem identificar estruturas mais complexas, possibilitando a classificação, a previsão ou a deteção de anomalias.

Comparativamente aos métodos tradicionais de *Machine Learning*, os modelos de *Deep Learning* demonstram um desempenho superior em casos que envolvam conjuntos de dados de elevada complexidade e dimensionalidade, como imagens, áudio e séries temporais. No entanto, modelos de *Deep Learning* requerem elevados volumes de dados para um treino eficaz e elevados requisitos computacionais, sendo comum o uso de *GPU*.

Este subconjunto de *Machine Learning* é amplamente utilizado em visão computacional, processamento de linguagem natural e análise preditiva. No contexto industrial, destaca-se particularmente na monitorização do estado dos equipamentos e na deteção de anomalias, permitindo identificar, de forma precoce, possíveis anomalias nas máquinas e aplicar estratégias de manutenção preditiva.

Deep Learning representa uma evolução significativa no *Machine Learning*, distinguindo-se pela capacidade de modelar relações complexas de forma automática e escalável, sendo atualmente uma das abordagens mais relevantes no desenvolvimento de sistemas inteligentes.

Neste projeto foram testados diversos modelos de *Deep Learning* com arquiteturas distintas.

²² <https://www.ibm.com/think/topics/deep-learning>, consultado em Maio de 2026

3.4.1 Autoencoder

Um Autoencoder é uma arquitetura de rede neural concebida para comprimir (codificar) dados de entrada de forma eficiente, reduzindo-os às suas características essenciais, e então reconstruir (descodificar) a entrada original a partir dessa representação comprimida.²³

Utilizando aprendizagem não supervisionada, os Autoencoders são treinados para descobrir variáveis latentes nos dados de entrada, ou seja, variáveis ocultas ou aleatórias que, apesar de não serem diretamente observáveis, influenciam fundamentalmente a forma como os dados são distribuídos.

Na figura seguinte está representada a arquitetura típica de um *Autoencoder*.

General Autoencoder Architecture

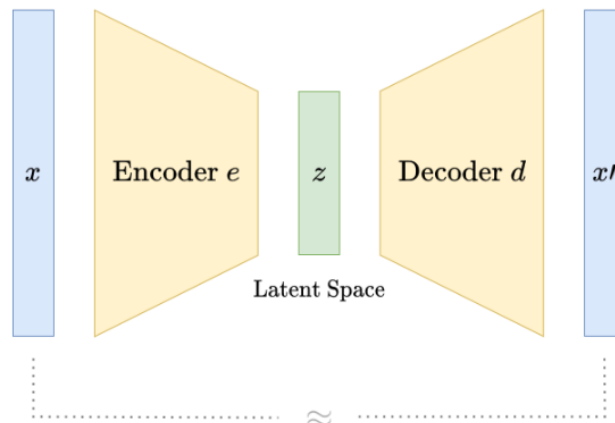


Figura 9 – Arquitetura Autoencoder²⁴

Os Autoencoders são fundamentalmente constituídos por 3 camadas e 2 fases:

Input Layer – Esta camada recebe os dados originais;

Hidden Layers – Estas camadas são responsáveis pela codificação e descodificação dos dados;

Output Layer – Esta camada mostra os dados de entrada reconstruídos pelas Hidden Layers;

Encoder – Nesta primeira fase o *encoder* reduz a dimensão dos dados de entrada, preservando os elementos mais significantes que melhor caracterizam os dados. Esta versão comprimida dos dados é passada pelo *Latent Space*;

²³ <https://www.ibm.com/think/topics/autoencoder>, consultado em Maio de 2026

²⁴ <https://towardsdatascience.com/introduction-to-autoencoders-b6fc3141f072/>, consultado em Maio de 2026

Decoder – Nesta fase são recebidos os dados comprimidos provenientes do *Latent Space* e são reconstruídos os dados originais através da sua representação comprimida, com o objetivo de minimizar a diferença entre entrada e saída.²⁵

A camada intermédia de *Hidden Layers* é onde se define o tipo de *Autoencoder*.

Model: "functional"

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 1024)	0
dense (Dense)	(None, 64)	65,600
dense_1 (Dense)	(None, 32)	2,080
dense_2 (Dense)	(None, 16)	528
dense_3 (Dense)	(None, 32)	544
dense_4 (Dense)	(None, 64)	2,112
dense_5 (Dense)	(None, 1024)	66,560

Total params: 137,424 (536.81 KB)
 Trainable params: 137,424 (536.81 KB)
 Non-trainable params: 0 (0.00 B)

Figura 10 – Modelo *Autoencoder*

Na figura acima está representado o sumário do *Autoencoder* utilizado no âmbito deste projeto, apenas com camadas *Dense* e uma arquitetura simétrica típica de um *Autoencoder* cuja função é aprender uma representação comprimida dos dados de entrada e depois reconstruir os mesmos.

Na camada de entrada, cada amostra possui 1024 características que vão diminuindo para 64, 32 e 16 nas camadas *Dense* seguintes. A camada “dense_2” representa o espaço latente, ou seja, a representação comprimida dos dados.

As camadas seguintes estão associadas ao *Decoder*, onde é feita a reconstrução dos dados a partir da sua representação comprimida. Como se trata de uma arquitetura simétrica, os dados reconstruídos são apresentados com 1024 características, ou seja, a dimensão original.

A aplicação ao caso de estudo de deteção de anomalias em sinais de vibração é conseguida através do erro de reconstrução, neste caso, MSE. Dados normais serão reconstruídos com erro reduzido, enquanto que dados de possíveis anomalias apresentarão um erro maior, uma vez que o modelo apresenta maior dificuldade em reconstruí-los por saírem do padrão esperado.

²⁵ <https://medium.com/@jagansaravana27/autoencoders-unsupervised-learning-using-neural-networks-f99bd9935b6c>, consultado em Maio de 2026

A classificação de anomalia ou funcionamento normal é conseguida através da comparação entre o MSE obtido na reconstrução e um valor de *Threshold* previamente definido. No caso do erro de reconstrução ser superior ao valor de threshold, é considerada anomalia.

O valor de *Threshold* utilizado em todos os modelos testados é obtido através da soma da média do erro de reconstrução a 3 vezes o desvio padrão do erro de reconstrução do modelo.

A figura seguinte representa o código *Python* para o cálculo do *Threshold*.

```
reconstruction_error = np.mean(np.square(X - X_pred), axis=1)

threshold = np.mean(reconstruction_error) + 3 * np.std(reconstruction_error)
anomalies = reconstruction_error > threshold
```

Figura 11 – *Threshold* e Classificação de Anomalia

3.4.2 Variational Autoencoder (VAE)

Tal como o Autoencoder, os VAE's são modelos de Deep Learning compostos por um encoder que aprende a isolar variáveis latentes importantes dos dados de treino e um decoder que usa essas mesmas variáveis latentes para reconstruir os dados de entrada.²⁶

No entanto, ao contrário de um Autoencoder tradicional que aprende uma representação fixa dos dados, o VAE aprende uma representação probabilística e contínua do espaço latente através da escolha de um ponto aleatório compreendido no intervalo dado pela média e pelo desvio padrão da distribuição latente, o que possibilita não só a reconstrução da entrada como também a produção de uma saída que se assemelha à entrada.²⁷

Na figura seguinte está representada a arquitetura típica de um VAE.

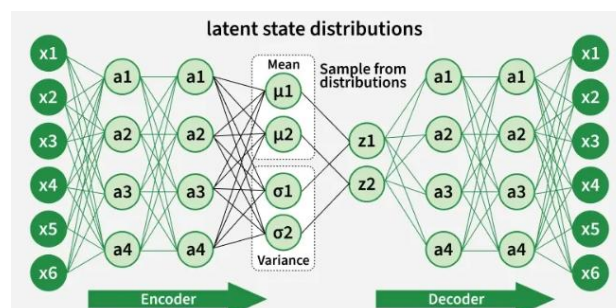


Figura 12 – Arquitetura Variational Autoencoder ²⁸

²⁶ <https://www.ibm.com/think/topics/variational-autoencoder>, consultado em Maio de 2026

²⁷ <https://www.geeksforgeeks.org/machine-learning/variational-autoencoders/>, consultado em Maio de 2026

²⁸ <https://www.geeksforgeeks.org/machine-learning/variational-autoencoders/>, consultado Maio de 2026

Model: "encoder"			
Layer (type)	Output Shape	Param #	Connected to
input_layer (InputLayer)	(None, 1024, 1)	0	-
flatten (Flatten)	(None, 1024)	0	input_layer[0][0]
dense (Dense)	(None, 256)	262,400	flatten[0][0]
dense_1 (Dense)	(None, 128)	32,896	dense[0][0]
z_mean (Dense)	(None, 16)	2,064	dense_1[0][0]
z_log_var (Dense)	(None, 16)	2,064	dense_1[0][0]
sampling (Sampling)	(None, 16)	0	z_mean[0][0], z_log_var[0][0]
Total params: 299,424 (1.14 MB)			
Trainable params: 299,424 (1.14 MB)			
Non-trainable params: 0 (0.00 B)			
Model: "decoder"			
Layer (type)	Output Shape	Param #	
input_layer_1 (InputLayer)	(None, 16)	0	
dense_2 (Dense)	(None, 128)	2,176	
dense_3 (Dense)	(None, 256)	33,024	
dense_4 (Dense)	(None, 1024)	263,168	
reshape (Reshape)	(None, 1024, 1)	0	
Total params: 298,368 (1.14 MB)			
Trainable params: 298,368 (1.14 MB)			
Non-trainable params: 0 (0.00 B)			

Figura 13 – Modelo Variational Autoencoder

A figura acima apresenta a arquitetura do *encoder* e *decoder* do VAE implementado no âmbito deste projeto. O *encoder* recebe uma entrada de 1024 amostras que é convertida num vetor unidimensional através da camada *flatten*. Após a conversão, são realizadas duas etapas de compressão até à dimensão de 128 amostras. As camadas *z_mean* e *z_log_var* são responsáveis pela produção de dois vetores de 16 amostras, sendo esta a etapa do espaço latente. A camada seguinte gera o vetor latente final *z* através da combinação entre a média e a variância, introduzindo variabilidade controlada no espaço latente que irá permitir ao modelo aprender distribuições mais generalizáveis.

O *decoder* reconstrói o sinal a partir do vetor latente, expandindo-o em três fases até retomar a dimensão original de 1024 amostras. Por fim, a camada *reshape* converte o vetor unidimensional na forma temporal original dos dados.

À semelhança do *Autoencoder*, a distinção entre eventos normais e eventos anómalos é feita através da comparação do erro de reconstrução com o valor de *threshold* obtido pela mesma fórmula.

3.4.3 Conv1D VAE

Este modelo consiste num VAE com a adição de camadas convolucionais unidimensionais.

As camadas convolucionais unidimensionais são projetadas especificamente para processar dados de sequências unidimensionais. Este tipo de camadas é particularmente útil para tarefas que envolvam sequências temporais onde os dados são inerentemente lineares e sequenciais, como é o caso de dados de vibração. A operação principal nestas camadas envolve o deslizamento de um filtro convolucional, ou *kernel*, ao longo da sequência de entrada. Esse filtro é essencialmente um conjunto de pesos treináveis que a rede ajusta durante o treino.

A operação de convolução multiplica os valores do *kernel* pelos valores de entrada originais num segmento da sequência, somando os resultados para produzir um único ponto de saída. Este processo é repetido em toda a sequência de entrada, gerando assim uma sequência de saída transformada.²⁹

Na figura abaixo está representado o modelo do Conv1D_VAE utilizado, onde é possível observar diversos parâmetros. Inicialmente, é feita uma normalização através do *BatchNorm1d* para tornar o formato dos dados no formato correto para o modelo; de seguida, são efetuadas operações convolucionais sucessivas, onde o modelo aprende padrões locais do sinal, e o *AvgPool1d* vai reduzindo a dimensão para metade da dimensão anterior até chegar à dimensão de 128 amostras. Após a extração de características, o *encoder* gera os vetores de média e desvio padrão e o tamanho do espaço latente é reduzido a uma representação probabilística de 10 dimensões em vez das 1024 amostras iniciais.

No *decoder*, é reconstruído o sinal original a partir da variável latente, onde, logo na primeira fase, é expandido para 512 amostras para preparar a reconstrução. As camadas de descodificação do sinal comprimido efetuam diferentes operações até chegar à representação de 1024 amostras iniciais. A função *Tanh* limita os valores reconstruídos ao intervalo $[-1,1]$, uma vez que o sinal tinha sido previamente normalizado.

A identificação de anomalias segue a lógica explanada anteriormente.

²⁹<https://www.geeksforgeeks.org/deep-learning/what-is-a-1d-convolutional-layer-in-deep-learning/>, consultado em Maio de 2026

Layer (type:depth-idx)	Output Shape	Param #
VAE	[1, 1]	--
└Encoder: 1-1	[1, 1]	--
└BatchNorm1d: 2-1	[1, 1, 1024]	2
└Conv_block: 2-2	[1, 32, 512]	--
└Conv1d: 3-1	[1, 32, 1024]	6,464
└BatchNorm1d: 3-2	[1, 32, 1024]	64
└ReLU: 3-3	[1, 32, 1024]	--
└AvgPool1d: 3-4	[1, 32, 512]	--
└Conv_block: 2-3	[1, 32, 256]	--
└Conv1d: 3-5	[1, 32, 512]	205,856
└BatchNorm1d: 3-6	[1, 32, 512]	64
└ReLU: 3-7	[1, 32, 512]	--
└AvgPool1d: 3-8	[1, 32, 256]	--
└Conv_block: 2-4	[1, 64, 128]	--
└Conv1d: 3-9	[1, 64, 256]	411,712
└BatchNorm1d: 3-10	[1, 64, 256]	128
└ReLU: 3-11	[1, 64, 256]	--
└AvgPool1d: 3-12	[1, 64, 128]	--
└Conv1d: 2-5	[1, 1, 129]	1,921
└AvgPool1d: 2-6	[1, 1, 1]	--
└AvgPool1d: 2-7	[1, 64, 64]	--
└Conv1d: 2-8	[1, 4, 64]	13,060
└Linear: 2-9	[1, 10]	2,570
└Linear: 2-10	[1, 10]	2,570
└Decoder: 1-2	[1, 1, 1024]	--
└Linear: 2-11	[1, 512]	5,632
└ReLU: 2-12	[1, 512]	--
└Conv_block: 2-13	[1, 64, 258]	--
└Conv1d: 3-13	[1, 64, 129]	51,264
└BatchNorm1d: 3-14	[1, 64, 129]	128
└ReLU: 3-15	[1, 64, 129]	--
└Upsample: 3-16	[1, 64, 258]	--
└Conv_block: 2-14	[1, 32, 518]	--
└Conv1d: 3-17	[1, 32, 259]	409,632
└BatchNorm1d: 3-18	[1, 32, 259]	64
└ReLU: 3-19	[1, 32, 259]	--
└Upsample: 3-20	[1, 32, 518]	--
└Conv_block: 2-15	[1, 32, 1038]	--
└Conv1d: 3-21	[1, 32, 519]	204,832
└BatchNorm1d: 3-22	[1, 32, 519]	64
└ReLU: 3-23	[1, 32, 519]	--
└Upsample: 3-24	[1, 32, 1038]	--
└Conv1d: 2-16	[1, 1, 1039]	6,401
└Tanh: 2-17	[1, 1, 1039]	--
Total params: 1,322,428		
Trainable params: 1,322,428		
Non-trainable params: 0		
Total mult-adds (Units.MEGABYTES): 444.18		
Input size (MB): 0.00		
Forward/backward pass size (MB): 1.60		
Params size (MB): 5.29		
Estimated Total Size (MB): 6.90		

Figura 14 – Modelo Conv1D_VAE

3.4.4 DAGMM

A detecção de anomalias é frequentemente realizada com recurso a técnicas de redução de dimensionalidade e agrupamento dos dados, no entanto, esses processos podem originar perda de informação relevante no processo de redução de dimensão, resultando em agrupamentos menos eficazes. Para mitigar este problema, o modelo em questão tem como objetivo otimizar os parâmetros de redução de dimensão e agrupamento, assegurando que o método não fica preso numa solução que não seja ótima, ajudando a reduzir ainda mais os erros de reconstrução.³⁰

O DAGMM considera conjuntamente um Deep Autoencoder e um *Gaussian Mixture Model* para modelar a distribuição de densidade de dados multidimensionais. O propósito do Deep Autoencoder é gerar uma representação de baixa dimensão e um erro de reconstrução para cada janela temporal dos dados de entrada, sendo posteriormente utilizada como entrada do GMM. Os parâmetros tanto do Deep Autoencoder quanto do GMM são otimizados simultaneamente de ponta a ponta, aproveitando uma rede de estimação separada para facilitar a aprendizagem dos parâmetros do GMM. Por fim, o DAGMM utiliza a probabilidade de observar as amostras de entrada como uma pontuação de anomalia. [18]

Ao contrário dos modelos vistos até ao momento, o DAGMM não utiliza a métrica MSE para determinar anomalias, utiliza um score que consiste na probabilidade de observar as amostras de entrada.

Na figura 15 está representada a arquitetura do DAGMM.

Para este projeto foi utilizado o repositório do GitHub.³¹

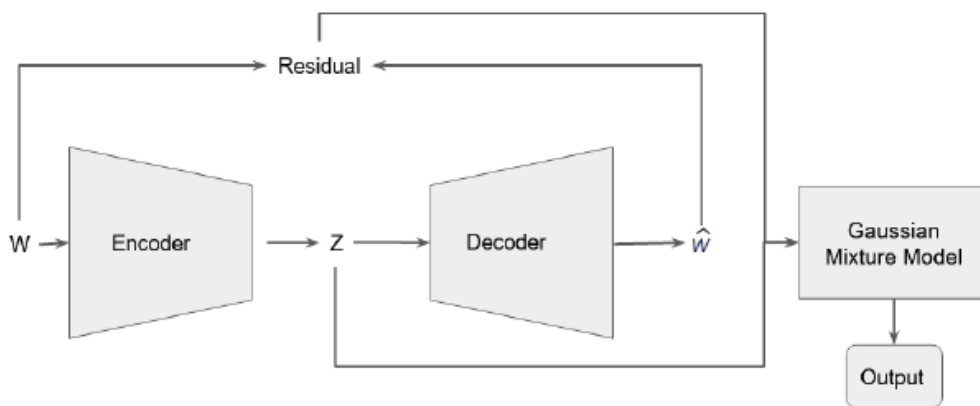


Figura 15 – Arquitetura DAGMM [18]

³⁰ <https://medium.com/@okpo65/dagmm-end-to-end-learning-with-autoencoder-and-gmm-for-anomaly-detection-28c687eac205>, consultado em Maio de 2026

³¹ <https://github.com/tnakae/DAGMM>, consultado em Maio de 2026

3.4.5 TCN-AE

A ideia central do TCN-AE é codificar a sequência de entrada numa sequência consideravelmente menor e subsequentemente reconstruir a sequência original a partir da sequência comprimida através de um decodificador. A ideia é similar à de um Autoencoder clássico, no entanto, diferencia-se no sentido em que substitui a arquitetura de camadas *dense* que se encontram em Autoencoders clássicos por uma arquitetura convolucional mais poderosa. [19]

Uma TCN é um tipo de arquitetura de rede neural projetada para processar dados sequenciais, como os dados de vibração, por exemplo. Ao contrário das RNNs que processam os dados passo a passo, as TCNs utilizam camadas convolucionais para capturar padrões ao longo do tempo, sendo a ideia principal aplicar convoluções unidimensionais sobre a sequência de dados de entrada, permitindo ao modelo aprender dependências temporais locais e globais de forma eficiente. Este tipo de arquitetura está representado na figura 16 e é particularmente eficaz para tarefas em que a ordem dos dados e o timing são importantes.

Uma característica que define estas redes é o uso de convoluções causais, que garantem que as previsões para um determinado instante dependam apenas de entradas passadas, e não de futuras. Esta característica evita fugas de dados e está alinhada com cenários de previsão em tempo real. Para capturar dependências de longo alcance, estas redes frequentemente incorporam convoluções dilatadas, onde os filtros saltam valores de entrada em intervalos controlados por um fator de dilatação, ou seja, um fator de dilatação igual a 2 significa que a operação de convolução abrange um instante de tempo sim, um não, ampliando efetivamente o campo recetivo da rede sem aumentar o número de parâmetros, que, numa aplicação *Edge*, é crucial.³²

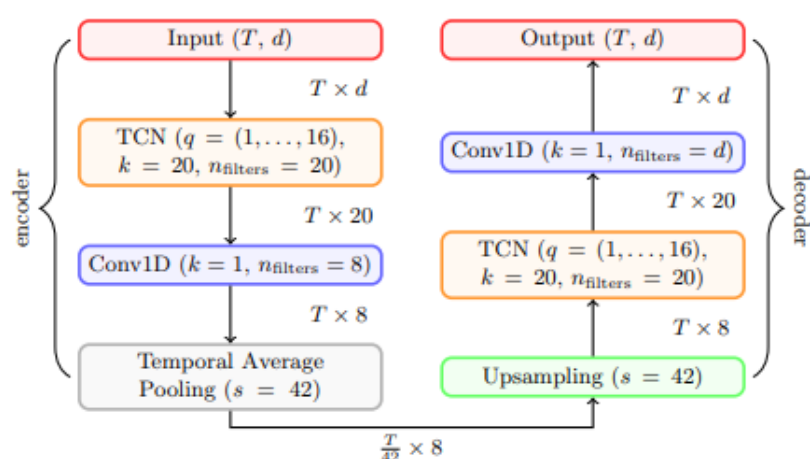


Figura 16 – Arquitetura TCN-AE [19]

³²<https://milvus.io/ai-quick-reference/what-are-temporal-convolutional-neural-networks>, consultado em Maio de 2026

O código utilizado é proveniente do repositório ³³ e a arquitetura acima demonstrada é referente a esse mesmo código, tendo sido retirada do artigo [19].

Model: "functional"

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, None, 1)	0
tcn-enc (TCN)	(None, None, 20)	72,640
conv1d (Conv1D)	(None, None, 8)	168
average_pooling1d (AveragePooling1D)	(None, None, 8)	0
activation (Activation)	(None, None, 8)	0
up_sampling1d (UpSampling1D)	(None, None, 8)	0
tcn-dec (TCN)	(None, None, 20)	75,580
dense (Dense)	(None, None, 1)	21

Total params: 148,409 (579.72 KB)
 Trainable params: 148,409 (579.72 KB)
 Non-trainable params: 0 (0.00 B)

Figura 17 – Modelo TCN-AE

Na figura acima encontra-se o sumário do modelo implementado onde são discriminadas as camadas constituintes do TCN-AE.

A camada *tcn-enc* extrai padrões temporais relevantes do sinal, extraindo 20 características de cada ponto temporal dos dados. Sobre essas 20 características extraídas, é aplicada uma convolução unidimensional para reduzir a dimensionalidade para 8, criando uma representação mais compacta dos dados. A camada seguinte, como o próprio nome indica, aplica uma média local para reduzir informação redundante, ajudando o modelo a focar-se nos padrões principais. de seguida, é aplicada uma função de ativação e a camada de *up_sampling1d* aumenta a dimensão dos dados de volta para as 20 características. A camada *tcn-dec* é responsável pela descodificação dos dados e reconstrução dos dados originais a partir da sua representação comprimida através de convoluções temporais. Na última camada é reposta a forma original dos dados, ou seja, uma característica por instante temporal.

³³ <https://github.com/MarkusThill/bioma-tcn-ae>, consultado em Maio de 2026

3.4.6 Transformer AE

Um Transformer Autoencoder tem como propósito aprender automaticamente padrões normais de um conjunto de dados através da combinação destes dois tipos de arquiteturas, onde o Autoencoder aprende a comprimir e reconstruir dados e o Transformer, através dos mecanismos de atenção, consegue perceber relações e dependências ao longo da sequência de dados.

As arquiteturas Transformer, representadas na figura 18, utilizam o mecanismo *Self-Attention* que permite que cada ponto do sinal olhe para os restantes pontos da sequência para perceber quais têm mais importância, quais as relações entre pontos distantes do sinal e perceber padrões temporais complexos, sendo bastante útil em séries temporais complexas.

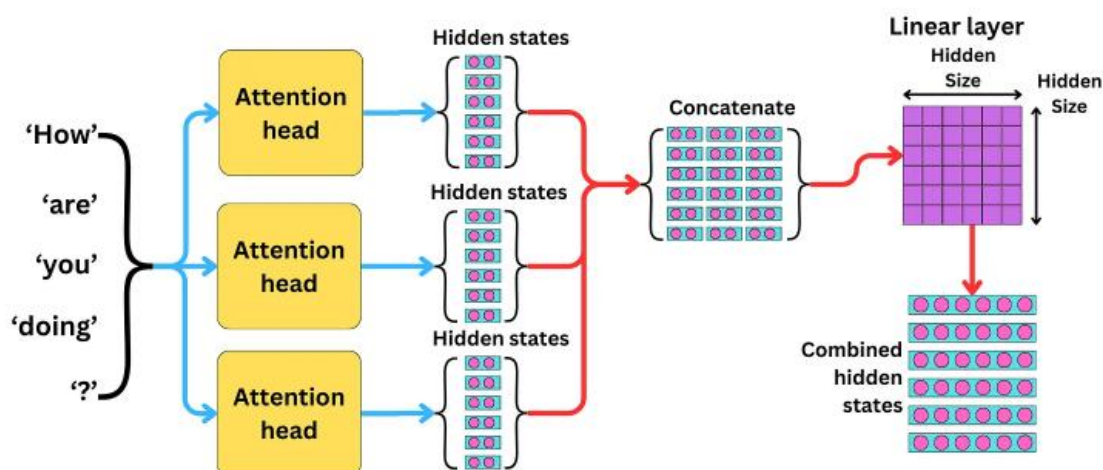


Figura 18 – Arquitetura Transformer AE ³⁴

³⁴ <https://newsletter.theaiedge.io/p/the-multi-head-attention-mechanism>, consultado em Maio de 2026

Model: "functional"

Layer (type)	Output Shape	Param #	Connected to
input_layer (InputLayer)	(None, 1024, 1)	0	-
dense (Dense)	(None, 1024, 32)	64	input_layer[0][0]
multi_head_attenti_ (MultiHeadAttentio...	(None, 1024, 32)	4,224	dense[0][0], dense[0][0]
dropout_1 (Dropout)	(None, 1024, 32)	0	multi_head_atten...
layer_normalization (LayerNormalizatio...	(None, 1024, 32)	64	dropout_1[0][0]
add (Add)	(None, 1024, 32)	0	layer_normalizat... dense[0][0]
conv1d (Conv1D)	(None, 1024, 64)	2,112	add[0][0]
dropout_2 (Dropout)	(None, 1024, 64)	0	conv1d[0][0]
conv1d_1 (Conv1D)	(None, 1024, 32)	2,080	dropout_2[0][0]
layer_normalization (LayerNormalizatio...	(None, 1024, 32)	64	conv1d_1[0][0]
add_1 (Add)	(None, 1024, 32)	0	layer_normalizat... add[0][0]
multi_head_attenti_ (MultiHeadAttentio...	(None, 1024, 32)	4,224	add_1[0][0], add_1[0][0]
dropout_4 (Dropout)	(None, 1024, 32)	0	multi_head_atten...
layer_normalization (LayerNormalizatio...	(None, 1024, 32)	64	dropout_4[0][0]
add_2 (Add)	(None, 1024, 32)	0	layer_normalizat... add_1[0][0]
conv1d_2 (Conv1D)	(None, 1024, 64)	2,112	add_2[0][0]
dropout_5 (Dropout)	(None, 1024, 64)	0	conv1d_2[0][0]
conv1d_3 (Conv1D)	(None, 1024, 32)	2,080	dropout_5[0][0]
layer_normalization (LayerNormalizatio...	(None, 1024, 32)	64	conv1d_3[0][0]
add_3 (Add)	(None, 1024, 32)	0	layer_normalizat... add_2[0][0]
dense_1 (Dense)	(None, 1024, 32)	1,056	add_3[0][0]
dense_2 (Dense)	(None, 1024, 1)	33	dense_1[0][0]

Total params: 18,241 (71.25 KB)
Trainable params: 18,241 (71.25 KB)
Non-trainable params: 0 (0.00 B)

Figura 19 – Modelo Transformer AE

A figura acima apresenta o sumário do modelo Transformer Autoencoder implementado, onde é possível observar que o modelo recebe sequências de 1024 pontos que são transformadas em 32 características diferentes para facilitar a aprendizagem. Depois, camadas *MultiHeadAttention*, típicas de Transformers, permitem ao modelo analisar relações entre diferentes partes do sinal através do mecanismo *Self-Attention*.

De seguida, as camadas Conv1D extraem padrões locais importantes, como picos ou oscilações do sinal. As camadas *Dropout* ajudam a evitar *overfitting* e as *LayerNormalization* melhoram a estabilidade do treino. As camadas *Add* criam ligações residuais que ajudam a preservar informação ao longo da rede.

A estrutura repete-se para melhorar a aprendizagem dos padrões do sinal e, no final, as camadas *Dense* reconstróem o sinal original.

3.4.7 USAD

O método USAD (UnSupervised Anomaly Detection) apresentado em [20] é formulado como uma arquitetura AE dentro de uma estrutura de treino adversarial de 2 fases. Por um lado, isto permite ultrapassar as limitações intrínsecas aos AEs ao treinar um modelo capaz de identificar quando os dados de entrada não contêm nenhuma anomalia e, ainda assim, reconstruir com qualidade; por outro lado, a arquitetura AE permite ganhos de estabilidade durante o treino adversarial, endereçando o problema de colapso e não-convergência encontrado em GANs.

USAD é composto por 3 elementos: um encoder (E) e dois decoders (D1, D2). Os 3 elementos são conectados a uma arquitetura composta por dois AEs, que partilham o mesmo encoder, e é treinada em duas fases. Primeiro, os dois AEs são treinados para aprender a reconstruir as janelas de dados normais e, em segundo, os dois AEs são treinados de forma adversarial, em que o AE1 vai tentar enganar o AE2 e o AE2 vai tentar aprender quando os dados são reais, vindos diretamente das janelas de dados ou reconstruídos pelo AE1.

Na primeira fase, o objetivo é treinar cada AE para reproduzir a entrada. As janelas de dados de entrada são comprimidas pelo encoder para o espaço latente e posteriormente reconstruídas por cada decoder.

Na segunda fase, o objetivo é treinar o AE2 a distinguir dados reais de dados provenientes do AE1 e treinar o AE1 para enganar o AE2. Dados provenientes do AE1 são comprimidos novamente pelo encoder para o espaço latente e depois reconstruídos pelo AE2. Usando uma configuração de treino adversarial, o objetivo do AE1 é minimizar a diferença entre as janelas de dados de entrada e a saída do AE2. O objetivo do AE2 é maximizar esta diferença. AE1 treina se é ou não sucedido em enganar o AE2 e o AE2 distingue os candidatos reconstruídos pelo AE1 dos dados reais.

Na figura 20 está representada a arquitetura USAD e respetivas fases de treino e deteção.

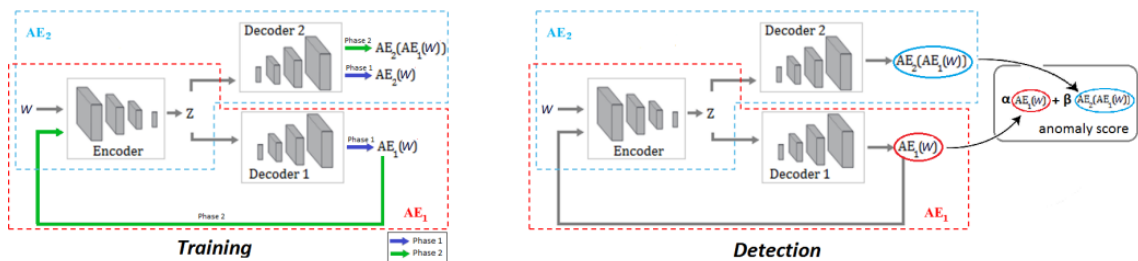


Figura 20 – Arquitetura USAD [20]

A implementação é dividida em três fases: uma primeira fase de pré-processamento dos dados comum em treino e deteção, onde os dados são normalizados e divididos em janelas, neste caso de 1024 dados; uma segunda fase para treinar o método offline e procurar capturar comportamentos normais de

porções pré-definidas e produzir um *anomaly score* para cada janela temporal; e a última fase é a detecção de anomalia que é executada online utilizando o modelo treinado na segunda fase. À chegada de uma nova janela temporal, o modelo é utilizado para obter um determinado *anomaly score*; se esse *anomaly score* for superior ao *threshold* definido, a janela é considerada anormal.

O *threshold* foi obtido da mesma forma em todos os modelos implementados.

Layer (type:depth-idx)	Output Shape	Param #
UsadModel	[1, 1024]	--
└Encoder: 1-1	[1, 128]	--
└└Linear: 2-1	[1, 512]	524,800
└└└ReLU: 2-2	[1, 512]	--
└└└Linear: 2-3	[1, 256]	131,328
└└└└ReLU: 2-4	[1, 256]	--
└└└└Linear: 2-5	[1, 128]	32,896
└└└└└ReLU: 2-6	[1, 128]	--
└Decoder: 1-2	[1, 1024]	--
└└Linear: 2-7	[1, 256]	33,024
└└└ReLU: 2-8	[1, 256]	--
└└└Linear: 2-9	[1, 512]	131,584
└└└└ReLU: 2-10	[1, 512]	--
└└└└Linear: 2-11	[1, 1024]	525,312
└└└└└Sigmoid: 2-12	[1, 1024]	--
└Decoder: 1-3	[1, 1024]	--
└└Linear: 2-13	[1, 256]	33,024
└└└ReLU: 2-14	[1, 256]	--
└└└Linear: 2-15	[1, 512]	131,584
└└└└ReLU: 2-16	[1, 512]	--
└└└└Linear: 2-17	[1, 1024]	525,312
└└└└└Sigmoid: 2-18	[1, 1024]	--

Figura 21 – Modelo USAD

Na figura acima está representado o sumário do modelo USAD implementado pelo repositório ³⁵, sendo possível observar o *encoder* e os dois *decoders*. O *encoder* recebe um vetor de entrada com 1024 valores que são reduzidos gradualmente até uma representação mais compacta de 128 valores, utilizando três camadas lineares sucessivas. Entre estas camadas são aplicadas funções de ativação *ReLU* que introduzem não linearidade no modelo e ajudam a aprender padrões mais complexos.

Após a compressão feita pelo *encoder*, os dois *decoders* tentam reconstruir os dados originais a partir da representação latente. Cada *decoder* faz o processo inverso ao *encoder*, aumenta gradualmente os dados de 128 para a forma original de 1024. No final, é utilizada uma função *Sigmoid* que limita os valores de saída entre 0 e 1.

³⁵ <https://github.com/manigalati/usad>, consultado em Maio de 2026

3.4.8 GRU

Gated Recurrent Units são um tipo de RNNs concebidas para lidar com dados sequenciais e ao mesmo tempo reduzir a complexidade das RNNs tradicionais. GRUs são um avanço simplificado de LSTMs, em que convergem múltiplos *gates* em *gates* de *update* e *reset*, aprendendo assim dependências de longo prazo com treino mais rápido e menos parâmetros.³⁶

A ideia principal por trás de GRUs é utilizar *gates* para atualizar seletivamente o estado oculto a cada passo de tempo, permitindo que se lembrem de informações importantes enquanto descartam detalhes irrelevantes. Comparativamente a LSTMs, GRUs acabam por simplificar a arquitetura uma vez que se focam em apenas dois *gates*.

Update Gate decide quanta informação do estado oculto anterior deve ser retida para o próximo passo temporal.

Reset Gate determina quanto do estado oculto do passado deve ser esquecido.

Estes dois mecanismos permitem que a GRU controle o fluxo de informação de uma forma mais eficiente em comparação com RNNs tradicionais, que dependem exclusivamente de estados ocultos.

Na figura 22 está representada a arquitetura GRU com os respetivos *gates*.

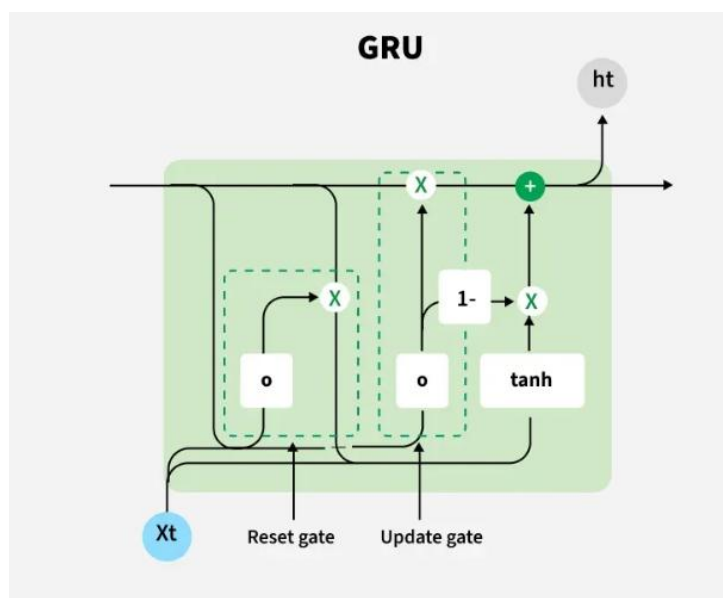


Figura 22 – Arquitetura GRU³⁷

³⁶ <https://www.geeksforgeeks.org/machine-learning/gated-recurrent-unit-networks/>, consultado em Maio de 2026

³⁷ <https://www.geeksforgeeks.org/deep-learning/rnn-vs-lstm-vs-gru-vs-transformers/>, consultado em Maio de 2026

Apesar de GRUs oferecerem um bom balanço entre desempenho e eficiência e igualarem ou até ultrapassarem LSTMs em determinadas tarefas, sendo mais rápidas e utilizando menos recursos, continuam a depender de processamento sequencial, o que limita a paralelização e prolonga o tempo de treino em sequências longas. Tal como LSTMs, podem ter dificuldades em lidar com dependências temporais de muito longo prazo.³⁸

Model: "functional_1"

Layer (type)	Output Shape	Param #
input_layer_4 (InputLayer)	(None, 1024, 1)	0
gru_14 (GRU)	(None, 1024, 64)	12,864
gru_15 (GRU)	(None, 32)	9,408
repeat_vector_3 (RepeatVector)	(None, 1024, 32)	0
gru_16 (GRU)	(None, 1024, 32)	6,336
gru_17 (GRU)	(None, 1024, 64)	18,816
time_distributed_2 (TimeDistributed)	(None, 1024, 1)	65

Total params: 47,489 (185.50 KB)
 Trainable params: 47,489 (185.50 KB)
 Non-trainable params: 0 (0.00 B)

Figura 23 – Modelo GRU

Como representado na figura 23, o modelo recebe vetores com 1024 instantes temporais e 1 variável, que é o valor de aceleração. As duas primeiras camadas GRU formam o *encoder*, extraindo características da sequência e comprimindo gradualmente até transformar a informação num vetor de 32 valores. A camada *RepeatVector* repete esse vetor ao longo dos 1024 instantes temporais para iniciar a reconstrução da sequência. Na parte do *decoder*, duas camadas GRU reconstroem os padrões temporais originais que, por fim, passam pela camada *TimeDistributed* que os distribui ao longo dos 1024 instantes temporais originais, repondo o formato original do vetor de entrada.

³⁸ <https://www.geeksforgeeks.org/deep-learning/rnn-vs-lstm-vs-gru-vs-transformers/>, consultado em Maio de 2026

3.4.9 LSTM

A arquitetura LSTM é um tipo especial de RNN concebida para aprender e relembrar informação ao longo de sequências de dados de maior dimensão.³⁹

Este tipo de arquitetura, representada na figura 24, utiliza uma célula de memória para guardar informação ao longo do tempo, resolvendo as limitações de RNNs tradicionais. É bastante aplicado em traduções linguísticas, reconhecimento de fala e previsões de séries temporais.⁴⁰

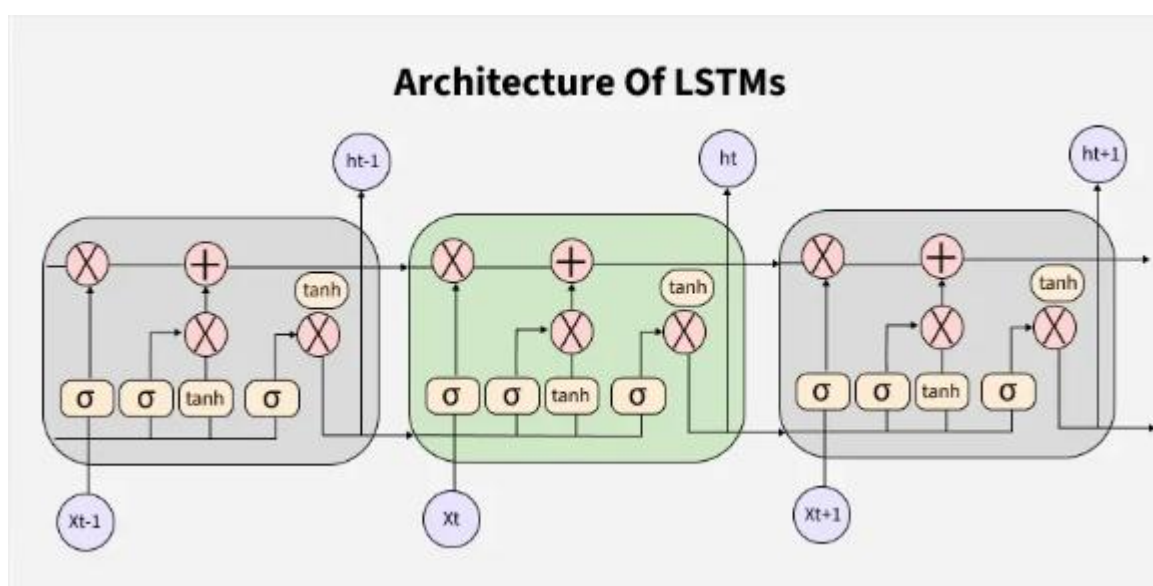


Figura 24 – Arquitetura LSTM

RNNs são concebidas para lidar com dados sequenciais através do uso de um estado oculto que guarda informação de passos anteriores. No entanto, demonstram dificuldade para aprender dependências de longo prazo, e isto acontece devido ao desaparecimento e à explosão do gradiente.

Ao treinar um modelo ao longo do tempo, os gradientes que ajudam o modelo a aprender podem encolher à medida que passam por muitos passos. Isto dificulta a aprendizagem de padrões a longo prazo, uma vez que a informação anterior se torna praticamente irrelevante.

Por vezes, os gradientes podem aumentar em demasia, causando instabilidade. Isto dificulta a aprendizagem adequada do modelo, uma vez que as atualizações se tornam erráticas e imprevisíveis.

³⁹ <https://www.ibm.com/think/topics/lstm>, consultado em Maio de 2026

⁴⁰ <https://www.geeksforgeeks.org/deep-learning/deep-learning-introduction-to-long-short-term-memory/>, consultado em Maio de 2026

No caso de LSTMs, são utilizados três *gates*, representados na figura 25:

Forget Gate – Determina quais informações devem ser removidas da célula de memória;

Input Gate – Decide qual nova informação deve ser adicionada à célula de memória;

Output Gate – Controla quais informações da célula de memória são passadas para o próximo estado oculto e para a saída.

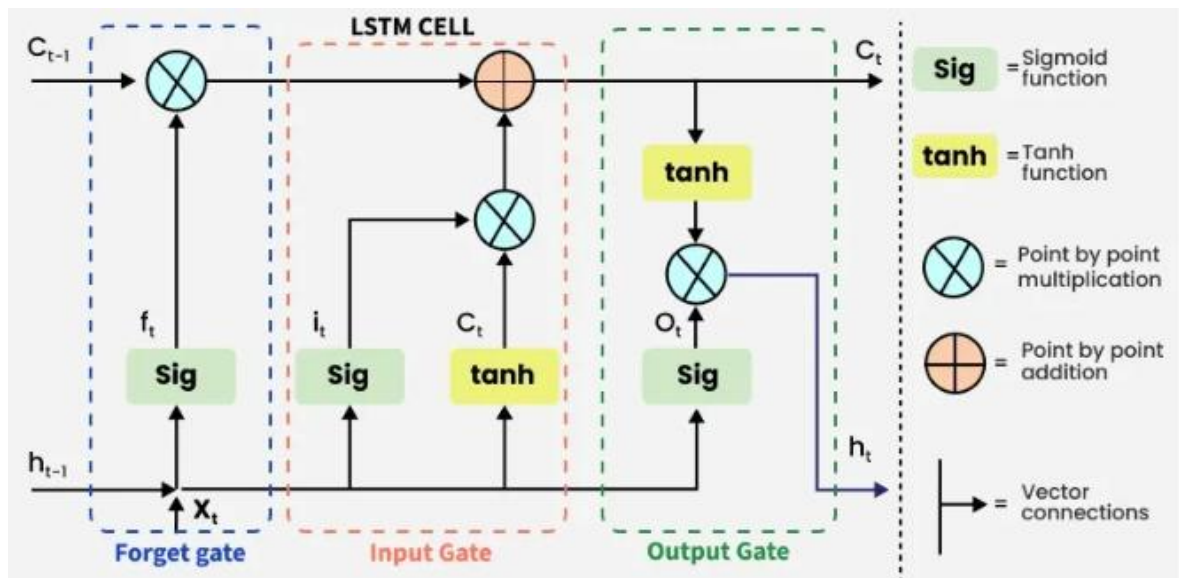


Figura 25 – LSTM Gates

No primeiro instante, o *Forget Gate* decide que informação deve ser mantida ou removida do estado da célula. O *gate* utiliza a entrada atual (x_t) e o estado oculto anterior (h_{t-1}), de seguida aplica uma função sigmoide para gerar valores entre 0 e 1, onde valores próximos de 0 removem informação e valores próximos de 1 retêm informação. Desta forma, controla a retenção de memória no LSTM e ajuda a descartar informações passadas desnecessárias.

De seguida, no *Input Gate*, a informação retida no passo anterior é regulada utilizando a função sigmoide e os valores a serem lembrados são filtrados de forma semelhante ao *Forget Gate*. Um vetor é criado usando a função $\tanh$, que fornece uma saída de -1 a 1, contendo todos os valores possíveis de h_{t-1} e x_t . Por fim, os valores do vetor e os valores regulados são multiplicados para obter as informações úteis.

No *Output Gate* é utilizado o estado oculto anterior (h_{t-1}) e a entrada atual (x_t), seguida por uma função sigmoide para controlar o fluxo de saída. Em seguida, o estado atual da célula é passado por uma ativação $\tanh$ para escalar os seus valores entre -1 e 1. Finalmente, este estado da célula transformado é multiplicado elemento a elemento pela ativação do *Output Gate* para produzir o estado oculto

h_t . Este estado oculto é então passado para o próximo passo de tempo e também pode ser utilizado para gerar a saída da rede.

Model: "functional"

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 1024, 1)	0
lstm (LSTM)	(None, 64)	16,896
repeat_vector (RepeatVector)	(None, 1024, 64)	0
lstm_1 (LSTM)	(None, 1024, 64)	33,024

Total params: 49,920 (195.00 KB)
 Trainable params: 49,920 (195.00 KB)
 Non-trainable params: 0 (0.00 B)

Figura 26 – Modelo LSTM

Na figura acima está representado o sumário do modelo AE LSTM implementado, recebendo janelas de 1024 dados com uma característica em cada instante. A primeira camada LSTM analisa toda a sequência e reduz a informação importante para um vetor de 64 características. Isto funciona como uma compressão dos dados, onde o modelo tenta guardar apenas os padrões mais relevantes da sequência original.

De seguida, a camada *RepeatVector* pega nesse vetor comprimido de 64 características e repete-o 1024 vezes, reconstruindo uma estrutura temporal com o mesmo tamanho da entrada. Por fim, a segunda camada LSTM utiliza essa informação para reconstruir a sequência original ao longo do tempo, produzindo uma saída com formato (1024, 64). Isto significa que, para cada 1024 instantes, a rede gera um conjunto de 64 valores internos que representam a informação aprendida naquele momento da sequência.

3.4.10 Transformers (TranAD)

Os *Transformers* são modelos populares de *Deep Learning* que têm sido usados em várias tarefas de processamento de linguagem natural e visão. No entanto, para o *TranAD*, os autores utilizam uma refatoração perspicaz da arquitetura *Transformer* para a tarefa de detecção de anomalias em dados de séries temporais. Assim como outros modelos *encoder-decoder*, em um *Transformer*, uma sequência de entrada passa por diversas transformações baseadas em atenção. A figura seguinte mostra a arquitetura da rede neural utilizada no *TranAD*.

O *encoder* codifica a sequência completa até ao *timestamp* atual (C) com uma pontuação de foco. O *encoder* de janela usa isso para criar uma representação

codificada da janela de entrada (W), que é então passada para dois *decoders* para criar a sua reconstrução. [21]

Na figura 27 está representada a arquitetura TranAD.

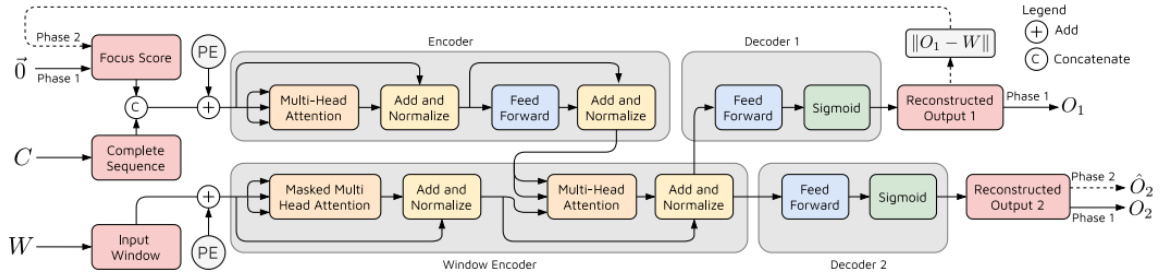


Figura 27 – Arquitetura TranAD

4. Resultados

4.1 Semáforo ISO/TR 17243-1

A simulação inicial contemplava uma janela na qual constavam o valor de vibração simulado num intervalo de 0 m/s² a 50 m/s², o carimbo de data/hora do registo, a classificação do respetivo valor e a cor de fundo, de acordo com a ISO/TR 17243-1. Na outra janela, apresentava-se um gráfico com os últimos 100 valores, cada um representado pela respetiva cor, conforme a norma.

Na figura abaixo, é possível observar os terminais onde estão a correr o *diagslave* e o *modpoll* para simular a comunicação em contexto real, bem como as duas janelas de saída do código, com valores aleatórios gerados a cada 2 segundos.

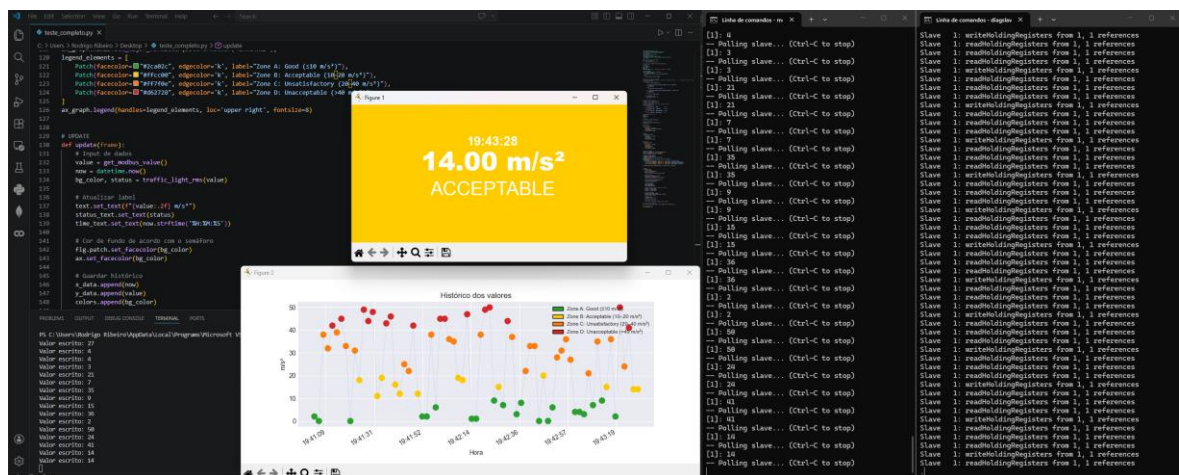


Figura 28 – Resultado da Simulação do Semáforo com Gráfico via *Modbus*

Na simulação final implementada na *HMI*, não consta o gráfico com os últimos 100 valores; para uma informação mais limpa, ficaram apenas o carimbo de data/hora do registo, o valor do registo em m/s², a classificação e o histórico dos 10 últimos registos, incluindo o atual.

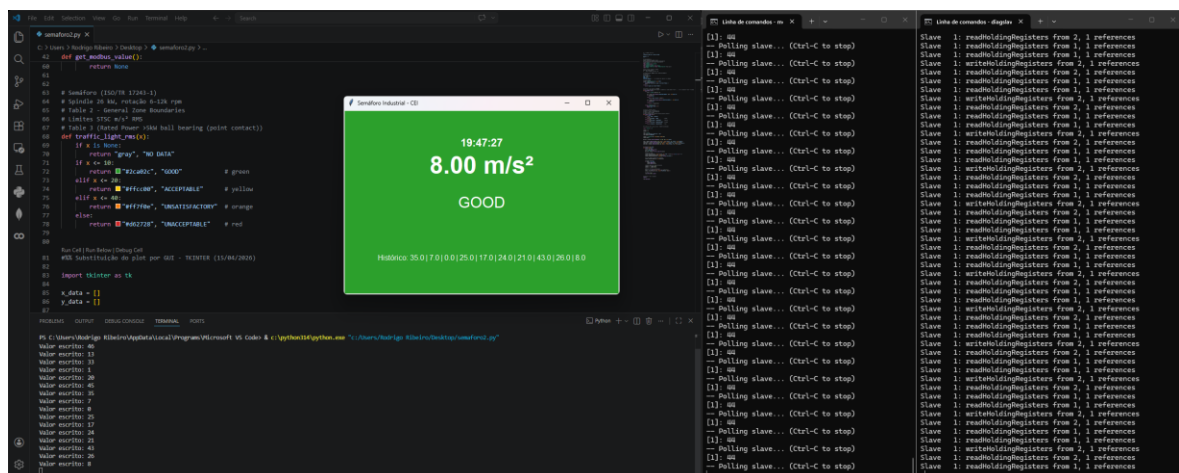


Figura 29 – Resultado da Simulação do Semáforo Implementado via *Modbus*

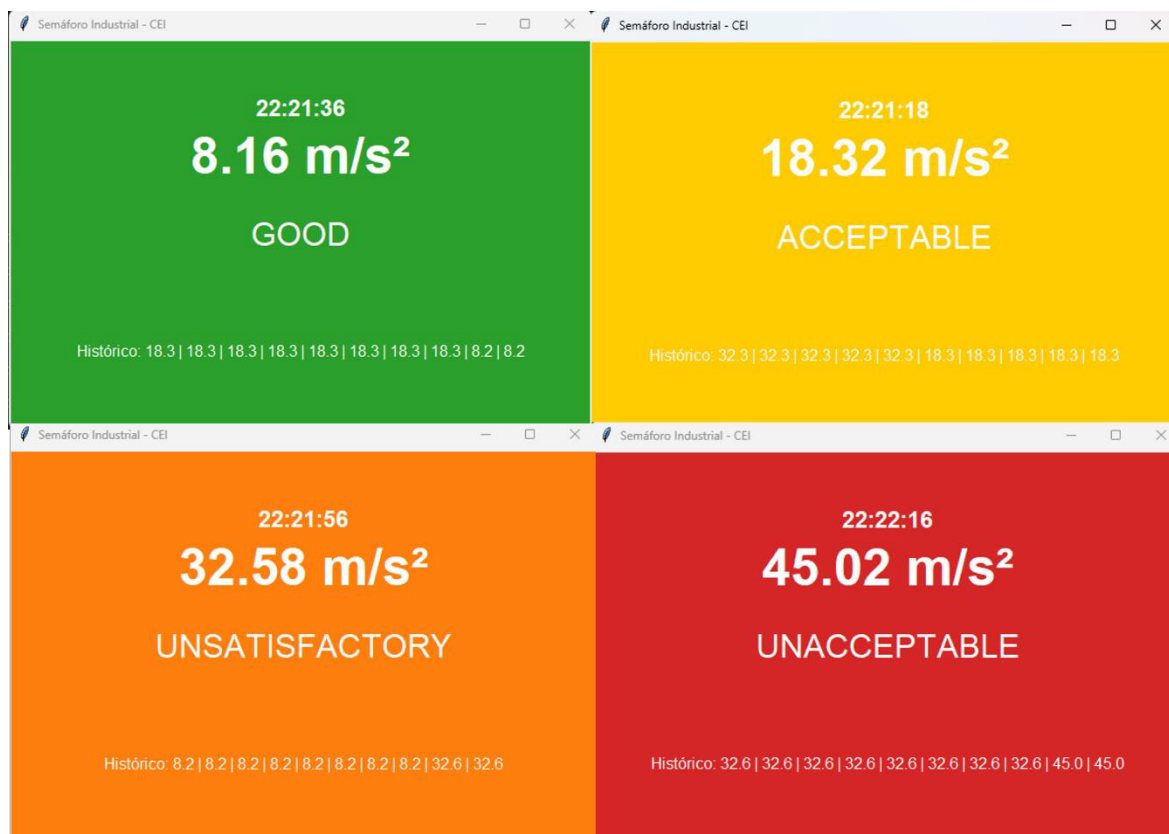


Figura 30 – Exemplificação do Semáforo Implementado

Na figura acima estão representadas as quatro janelas possíveis do semáforo, de acordo com a ISO/TR 17243-1.

Como são cores bastante distintas e bem evidentes, captam rapidamente a atenção do operador e são de rápida interpretação. Na eventualidade do operador apenas reparar na cor e, por algum motivo, não reparar no valor, o mesmo fica disponível no histórico.

A figura 31 ilustra a implementação do semáforo industrial aplicado na máquina.



Figura 31 – StoneCut com Semáforo na Solancis

No equipamento, o semáforo apresenta o valor atual com a respetiva *timestamp*, cor de fundo de acordo com a norma ISO/TR 17243-1 e, no histórico, os últimos seis valores registados.

Nesta visita foi possível obter *feedback* da parte de um operador do equipamento acerca do semáforo, tendo sido dito pelo mesmo que, até à data, só tinha observado fundo verde e amarelo, o que é um dado bastante importante para a justificação da parte seguinte de deteção de anomalias. Na figura seguinte está representado o semáforo com maior detalhe.



Figura 32 – StoneCut com Semáforo na Solancis (Detalhe)

4.2 Detecção de Anomalias

Para a secção de detecção de anomalias através de modelos de *Machine Learning* e *Deep Learning*, foram testados treze algoritmos em três *datasets* distintos, dos quais, entre os diversos recolhidos nas instalações e por acesso remoto ao equipamento, dois deles apresentavam picos bastante evidentes e o terceiro um comportamento mais estável.

Para cada modelo foi gerado um gráfico onde constam as anomalias detetadas pelo mesmo, assinaladas com um retângulo vermelho, e indicadores de desempenho relevantes para avaliar a sua aplicabilidade em *Edge Computing*.

As métricas utilizadas estão divididas entre avaliadores de aplicabilidade em *Edge Computing* e avaliadores de capacidade de detecção de anomalias de cada modelo.

O tempo de treino representa, em segundos, a duração necessária para ajustar cada modelo ao conjunto de dados de vibração durante a fase de desenvolvimento. Apesar de o treino poder ser realizado num outro computador com maior capacidade de processamento e posteriormente transferir os resultados desse treino para o *Edge computer*, esta métrica permite avaliar a complexidade do processo de aprendizagem e a facilidade de efetuar um novo treino caso necessário. A necessidade de realizar um novo treino pode ser devida a várias circunstâncias. Pode ter sido efetuada uma grande reparação e é recomendado realizar um novo treino com dados recolhidos após essa reparação para aprender o novo comportamento do equipamento. No caso de ser adicionada uma ferramenta com um comportamento específico, também é aconselhável treinar o modelo com dados recolhidos durante a utilização dessa mesma ferramenta, etc. Desta forma, é vantajoso adotar um modelo com um tempo de treino baixo, pois irá facilitar e acelerar a adaptação do sistema de detecção de anomalias a novas condições de funcionamento da máquina.

O tempo total de inferência, representado em milissegundos, corresponde ao tempo necessário para o modelo analisar todas as janelas de dados de vibração e gerar os respetivos *scores* de anomalia. Esta métrica é fundamental para implementação em *Edge Computing*, uma vez que o processamento é realizado no local da aquisição dos dados. Tempos de inferência reduzidos permitem uma detecção de anomalias mais rápida.

O tempo de inferência por amostra representa, em milissegundos, o tempo médio necessário para processar cada amostra, ou seja, cada janela de 1024 valores de vibração. Esta métrica é uma das mais relevantes para a implementação em *Edge*, pois permite estimar a latência do sistema durante a monitorização contínua do equipamento. Um menor tempo de inferência por amostra significa que o modelo consegue acompanhar maiores frequências de aquisição de dados ou até mesmo analisar múltiplos equipamentos utilizando o mesmo dispositivo *Edge*. Para este caso em particular, os dados foram recolhidos a uma frequência de 50 kHz, ou

seja, 50 000 dados por cada segundo de recolha. Como foram sempre utilizadas janelas de 1024 valores, equivale a aproximadamente 49 amostras, ou seja, janelas de 1024 valores, por cada segundo de recolha de dados. Para estes valores, através da equação (1), sabe-se que o sensor adquire um novo valor de vibração a cada 20 μs e, através da equação (2), sabe-se que cada amostra representa, aproximadamente, 20,48 ms do sinal de vibração.

$$T_{valor} = \frac{1}{f} \leftrightarrow T = \frac{1}{50\,000} \leftrightarrow T_{valor} = 20\,\mu s \quad (1)$$

$$T_{janela} = \frac{1024}{50\,000} \leftrightarrow T_{janela} = 20,48\,ms \quad (2)$$

Uma vez calculados estes tempos, para uma inferência em tempo real em *Edge*, é necessário um modelo com um tempo de inferência por amostra igual ou inferior a 20,48 ms.

Passando para as métricas que avaliam a capacidade de deteção de anomalias, é comparado o *threshold*, no entanto, não é a métrica mais relevante porque, apesar de ter sido utilizado o mesmo critério de definição de *threshold* para todos os modelos, modelos como USAD, DAGMM, COPOD, ECOD e HBOS possuem uma métrica própria para definir o limiar de separação entre assinalar um dado como normal ou anómalo, não podendo comparar diretamente com o *threshold* dos restantes modelos.

As métricas *MSE Mean* e *MSE Std.* são referentes à média e ao desvio padrão do erro quadrático médio. A média do erro quadrático médio representa o erro médio de reconstrução obtido pelo modelo para as janelas analisadas e o desvio padrão do erro quadrático médio representa a variabilidade do erro de reconstrução entre diferentes janelas de vibração. Como foram apenas utilizados modelos de aprendizagem não supervisionada por não existirem rótulos nos dados, os modelos aprendem o comportamento normal do equipamento e tentam reconstruir sinais de vibração. Quando um padrão diferente surge nos dados, o erro de reconstrução tende a aumentar porque a saída produzida pelo modelo apresenta mais diferenças em relação à entrada. Uma média do erro quadrático médio mais elevada significa que o modelo apresentou maior dificuldade em reconstruir corretamente os dados de entrada, podendo significar uma maior presença de dados anómalos. Um valor mais elevado do desvio padrão do erro quadrático médio pode indicar que determinadas condições do funcionamento do equipamento foram mais difíceis de

representar ou que existem variações no sinal que podem estar relacionadas a eventos anômalos, enquanto que um valor mais baixo do desvio padrão indica que o modelo apresenta um comportamento estável durante a monitorização.

A *kurtosis* caracteriza a presença de picos ou eventos impulsivos no sinal de vibração. Em sistemas mecânicos, como o equipamento em questão, aumentos de *kurtosis* estão frequentemente associados a fenômenos como impactos, desgaste ou defeitos localizados, enquanto que um valor de *kurtosis* mais baixo indica que o sinal tem uma distribuição com menos valores extremos, ou seja, um sinal mais estável e, à partida, sem anomalias. Esta métrica é útil para complementar os modelos de deteção, permitindo assim avaliar características do sinal juntamente com a classificação realizada pelo modelo.

A *skewness* representa a assimetria da distribuição dos valores do sinal de vibração e permite identificar alterações na forma do sinal relativamente ao comportamento normal. Valores próximos de zero indicam uma distribuição equilibrada, enquanto que valores elevados indicam maior concentração de valores extremos numa determinada direção, permitindo desta forma diferenciar padrões e condições de operação normais.

A utilização da distância de *Mahalanobis* adveio de [11], onde os autores utilizaram a distância de *Mahalanobis* em vez da distância euclidiana, alegando que a distância euclidiana não contabiliza a variabilidade e correlação entre os dados. A distância de *Mahalanobis*, ao contrário da distância euclidiana, permite avaliar o desvio entre o sinal medido e o sinal previsto tendo em conta o comportamento estatístico esperado. *Mahalanobis Mean* representa o valor médio da distância calculada para todas as janelas de vibração analisadas, indicando o afastamento médio entre os valores previstos pelo modelo e os valores realmente medidos, considerando a variabilidade e correlação entre os dados. *Mahalanobis Std.* representa a dispersão dos valores da distância entre as diferentes janelas, permitindo avaliar a consistência dos erros de previsão ao longo da monitorização. Um modelo robusto num conjunto de dados normal deve apresentar valores, tanto de média como de desvio padrão, reduzidos, indicando assim uma boa capacidade de previsão do comportamento. Valores elevados destas métricas indicam maiores desvios e maior variabilidade, podendo representar condições anômalas ou menor estabilidade do modelo. Como a distância de *Mahalanobis* é calculada levando em consideração o valor real e o valor previsto, modelos como COPOD, ECOD e HBOS não possuem valor de distância porque não fazem previsão, fazem a deteção de anomalias apenas baseada na distribuição dos dados.

Todos os testes foram realizados no Google Colab utilizando *GPU T4*, à exceção do LSTM, que, por demorar aproximadamente duas horas a treinar o modelo com a *GPU T4*, foi utilizada a *v6e-1 TPU* também do Colab.

De seguida vão ser discutidos os resultados obtidos pelos treze modelos nas métricas mencionadas em três *datasets* diferentes.

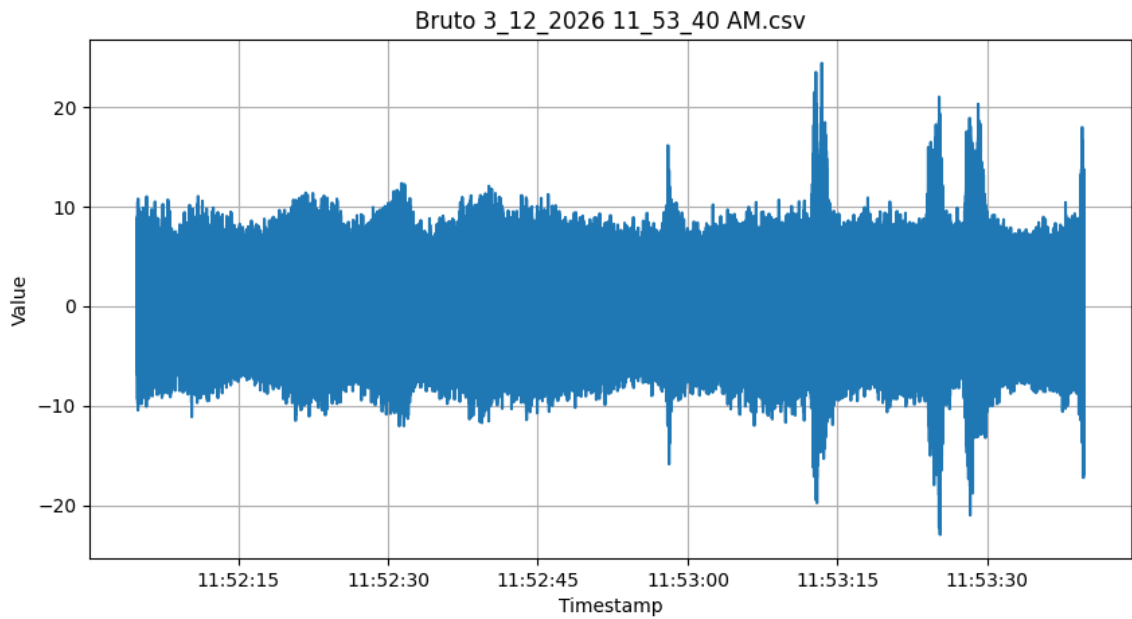


Figura 33 – Dataset 1

A figura 33 representa o *dataset* 1 que foi recolhido no dia 12 de Março de 2026, a uma frequência de amostragem de 50 kHz durante cerca de 1 minuto e meio. À primeira vista, é possível identificar cinco picos mais significativos no valor de vibração expresso em m/s^2 .

Tabela 3 – Resultados das Métricas de Avaliação Dataset 1

Metrics	Autoencoder	ConvID VAE	COPOD	DAGMM	ECOD	GRU RNN	HBOS	LSTM	TCN AE	TranAD	Transformer AE	USAD	VAE
Training Time (s)	21,18	220,47	2,23	9,58	2,28	1683,47	4,84	834,70	210,57	218,83	161,93	21,27	29,31
Total Inference Time (ms)	2649,83	1156,16	3053,94	91,27	3205,86	26858,98	458,08	15367,42	5112,20	3644,15	4711,46	87,94	2256,99
Inference per Sample (ms)	0,57	0,25	0,66	0,02	0,69	5,80	0,10	3,32	1,10	0,79	1,27	0,02	0,61
Threshold	1,66	2,51	1992,09	93,68	2574,62	2,51	1928,22	2,51	0,29	1,73	0,0040	2,51	1,10
MSE Mean	0,44	1,00	1373,35	46,66	1726,92	1,00	1534,12	1,00	0,07	0,62	0,0005	1,00	0,34
MSE Std.	0,41	0,50	206,25	15,67	282,56	0,51	131,37	0,50	0,07	0,37	0,0011	0,50	0,25
Kurtosis	30,37	16,67	8,16	77,54	7,20	16,62	4,21	16,66	47,65	21,20	378,51	16,66	78,07
Skewness	5,17	3,46	2,34	7,89	2,20	3,46	1,66	3,46	6,08	3,97	17,51	3,46	8,11
Mahalanobis Mean	3,38	2,90	NaN	5,81	NaN	5,07	NaN	3,80	15,34	0,78	0,82	9,18	3,93
Mahalanobis Std.	0,76	1,15	NaN	0,50	NaN	2,35	NaN	2,48	4,49	0,62	0,56	1,66	0,76
Device	GPU T4	GPU T4	GPU T4	GPU T4	GPU T4	GPU T4	GPU T4	TPU v6e-1	GPU T4	GPU T4	GPU T4	GPU T4	GPU T4

Na tabela acima constam os resultados obtidos por cada modelo durante a análise do *dataset* 1.

Os modelos baseados em redes neurais apresentam, em geral, maiores tempos de treino devido à sua complexidade, destacando-se o GRU RNN (1683,47 s) e o LSTM (834,70 s). Em contraste, métodos estatísticos como COPOD, ECOD e HBOS apresentam tempos de treino bastante inferiores, sendo mais simples de atualizar. No entanto, estes métodos não realizam reconstrução do sinal, pelo que métricas como *MSE* e *Mahalanobis* não são diretamente comparáveis, apresentando valores elevados ou *NaN*.

Relativamente ao desempenho em execução, a métrica *Inference per Sample* é essencial para avaliar a aplicação em *Edge*. Considerando a frequência de aquisição de 50 kHz e janelas de 1024 amostras, cada janela corresponde a aproximadamente 20,48 ms, sendo esse o tempo máximo disponível para processamento em tempo real com uma taxa de aquisição de dados de 50 kHz.

Todos os modelos apresentam tempos inferiores a este limite, destacando-se o USAD (0,02 ms), DAGMM (0,02 ms) e HBOS (0,10 ms).

Na avaliação do erro de reconstrução, o *Transformer AE* apresenta o menor *MSE Mean* (0,0005) e *MSE Std* (0,0011), indicando uma elevada capacidade de reconstrução e grande estabilidade. O TCN AE também apresenta bons resultados, com baixo erro médio e reduzida variabilidade.

As métricas estatísticas do sinal, *kurtosis* e *skewness*, permitem caracterizar alterações no comportamento da vibração. Valores elevados destas métricas indicam maior presença de picos e assimetrias no sinal, podendo estar associados a eventos anómalos. Apesar do *Transformer AE* apresentar valores muito baixos de *MSE*, apresenta também valores extremamente altos na *kurtosis* e na *skewness*, indicando uma sensibilidade excessiva ao ruído, que pode gerar falsos alarmes em contexto industrial real.

A distância de Mahalanobis permite avaliar o afastamento entre os valores previstos e os valores medidos considerando as correlações dos dados. Valores mais baixos indicam previsões mais próximas do comportamento esperado. Os melhores resultados foram obtidos pelo TranAD (0,78) e pelo Transformer AE (0,82), indicando menor desvio estatístico.

De forma geral, os resultados mostram que modelos como o Transformer AE apresentam melhor desempenho na reconstrução dos sinais, mas também alguma sensibilidade que pode prejudicar a deteção em contexto industrial, enquanto modelos como USAD e DAGMM apresentam vantagens em termos de velocidade de inferência. Assim, a escolha do modelo para *Edge* deve considerar o equilíbrio entre precisão, estabilidade e custo computacional.

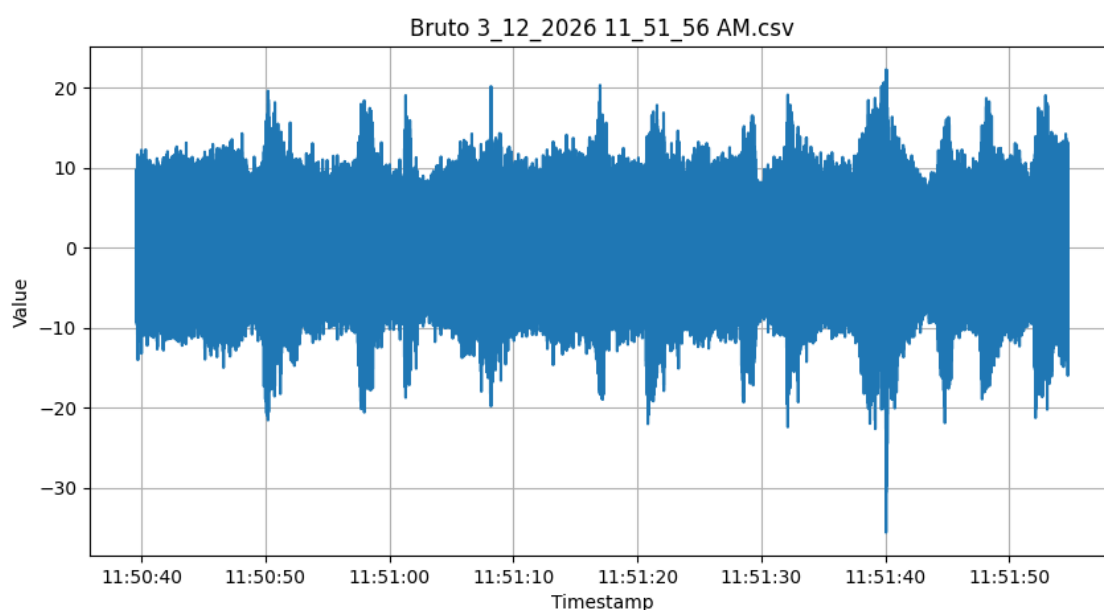


Figura 34 – Dataset 2

Na figura 34 encontra-se representado o *dataset* 2 que foi recolhido no mesmo dia que o *dataset* 1, com a mesma frequência de amostragem, durante, aproximadamente, 1 minuto. Apresenta diversos picos no sinal medido.

Tabela 4 – Resultados das Métricas de Avaliação Dataset 2

Metrics	Autoencoder	ConvID VAE	COPOD	DAGMM	ECOD	GRU RNN	HBOS	LSTM	TCN AE	TranAD	Transformer AE	USAD	VAE
Training Time (s)	17,09	181,77	1,83	9,12	4,01	1441,17	3,60	1536,81	193,82	169,38	273,99	18,59	31,14
Total Inference Time (ms)	1111,24	949,22	2878,50	79,39	4469,88	24576,56	246,17	12307,73	5437,16	2841,98	4192,44	73,89	1795,90
Inference per Sample (ms)	0,30	0,26	0,78	0,02	1,22	6,70	0,07	3,36	1,48	0,78	1,43	0,02	0,61
Threshold	1,47	2,46	2041,44	58,89	2663,68	2,18	199,38	2,46	0,10	2,46	0,0003	2,46	0,90
MSE Mean	0,37	1,00	1374,05	39,33	1727,73	0,87	1526,12	1,00	0,04	1,00	0,000051	1,00	0,28
MSE Std.	0,37	0,49	222,46	6,52	311,98	0,44	155,42	0,49	0,02	0,49	0,000079	0,49	0,21
Kurtosis	17,90	10,77	2,39	4,38	1,93	10,54	0,50	10,78	25,05	11,18	273,87	10,78	7,90
Skewness	3,17	2,04	1,10	1,81	1,01	2,21	0,68	2,04	3,69	2,07	13,49	2,04	2,59
Mahalanobis Mean	3,39	2,82	NaN	5,76	NaN	4,64	NaN	3,56	15,48	0,80	0,62	9,08	3,85
Mahalanobis Std.	0,69	1,37	NaN	0,89	NaN	2,32	NaN	2,62	4,01	0,60	0,42	1,89	1,08
Device	GPU T4	GPU T4	GPU T4	GPU T4	GPU T4	GPU T4	GPU T4	TPU v6e-1	GPU T4	GPU T4	GPU T4	GPU T4	GPU T4

Os resultados obtidos em termos de tempos de treino e inferência mantêm a lógica observada no *dataset* anterior, não com os mesmos valores por ser um *dataset* de dimensão diferente, mas os modelos mais rápidos e mais lentos continuam os mesmos.

Todas as conclusões retiradas no *dataset* anterior replicam-se neste. Os modelos tiveram comportamentos semelhantes nestes dois *datasets* com picos.

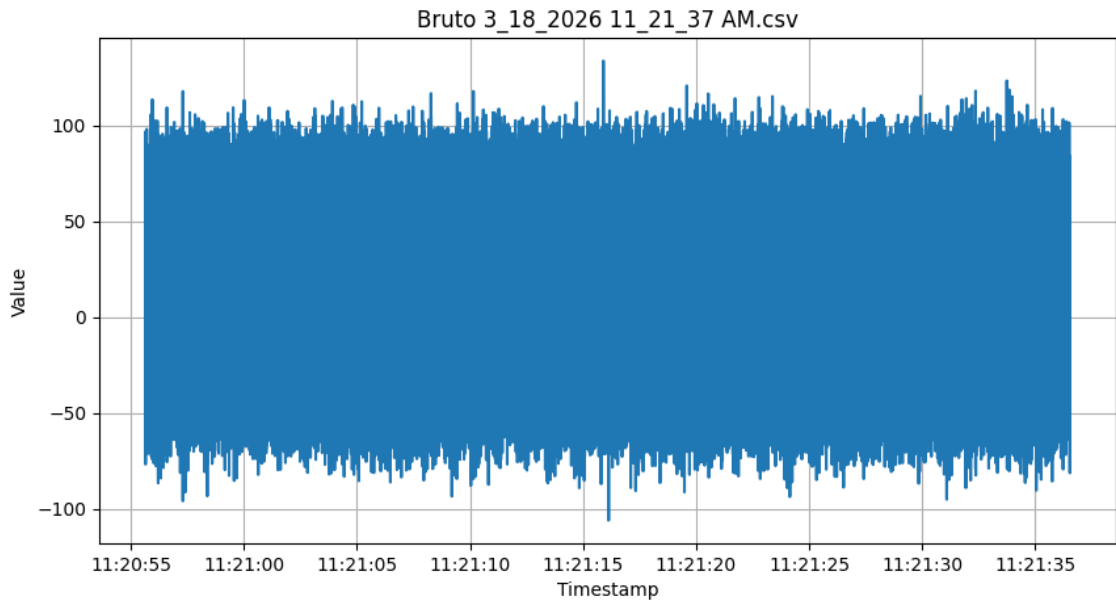


Figura 35 – Dataset 3

O terceiro e último *dataset* analisado, representado na figura 35, foi recolhido no dia 18 de Março de 2026, presencialmente, durante aproximadamente 40 segundos. Apesar de, dos 3 *datasets*, este ser aparentemente o mais estável, é importante notar que a amplitude do sinal de vibração vai de -100 m/s² a 100 m/s². Nos *datasets* anteriores, o sinal variava entre -30 m/s² e 20 m/s². Para neste caso existirem valores tão elevados de vibração sem anomalia real no equipamento, este *dataset* pode ter sido recolhido durante o processo de corte de pedra com disco, o que explica estes valores.

Tabela 5 – Resultados das Métricas de Avaliação Dataset 3

Metrics	Autoencoder	Conv1D VAE	COPOD	DAGMM	ECOD	GRU RNN	HBOS	LSTM	TCN AE	TranAD	Transformer AE	USAD	VAE
Training Time (s)	12.81	95.61	1.43	6.98	1.43	181.27	5.92	479.95	124.67	100.11	161.21	13.85	20.25
Total Inference Time (ms)	1779.51	544.13	1279.89	71.05	1189.26	19404.93	103.34	6764.83	5504.61	1748.41	3225.86	69.24	2248.98
Inference per Sample (ms)	0.89	0.27	0.64	0.04	0.60	9.72	0.05	3.39	2.76	0.88	2.02	0.03	1.41
Threshold	1.02	1.38	1569.21	50.86	1922.94	1.34	1561.47	1.38	0.49	0.75	0.0014	1.38	1.04
MSE Mean	0.76	1.00	1374.87	42.85	1728.57	1.00	1425.57	1.00	0.37	0.60	0.000403	1.00	0.78
MSE Std.	0.08	0.13	64.78	2.67	64.79	0.13	45.30	0.13	0.04	0.05	0.000336	0.13	0.08
Kurtosis	0.07	0.03	0.04	2.59	-0.07	0.03	-0.02	0.04	0.09	0.12	96.16	0.04	0.12
Skewness	0.29	0.33	0.30	0.99	0.18	0.33	0.16	0.33	0.23	0.27	7.70	0.33	0.36
Mahalanobis Mean	3.62	2.97	NaN	5.78	NaN	2.94	NaN	3.61	15.95	0.80	0.83	9.13	3.94
Mahalanobis Std.	0.95	1.07	NaN	0.51	NaN	1.67	NaN	1.99	1.01	0.61	0.59	1.63	0.67
Device	GPU T4	GPU T4	GPU T4	GPU T4	GPU T4	GPU T4	GPU T4	TPU v6e-l	GPU T4	GPU T4	GPU T4	GPU T4	GPU T4

Em relação a tempos de treino e inferência, a lógica mantém-se, as principais diferenças aparecem nas métricas *kurtosis* e *skewness*.

Praticamente todos os modelos apresentaram valores de *kurtosis* e *skewness* próximos de zero, o que significa que a distribuição dos dados é equilibrada e não apresenta picos significativos, o que, olhando para o gráfico dos dados em bruto, faz sentido.

A distância de *Mahalanobis* não sofreu grandes alterações, houve alguns modelos que apresentaram uma distância média mais baixa, o que significa que a distância, em média, entre o valor previsto e o valor real diminuiu, mas a maior diferença foi no desvio padrão da distância que baixou consideravelmente em praticamente todos os modelos em comparação com os valores dos outros *datasets*, o que significa que houve menos variabilidade nas distâncias ao longo do *dataset*.

Estes valores estão de acordo com o gráfico observado, não havendo picos significativos como anteriormente. *Kurtosis* e *skewness* devem ficar próximos de zero e a distância de Mahalanobis também, no entanto, a distância de Mahalanobis não depende apenas do sinal analisado, depende também da eficácia de cada modelo em reproduzir corretamente o sinal de entrada.

De notar que, o *Transformer AE*, foi o único modelo que apresentou *kurtosis* e *skewness* diferentes de zero. Esta informação valida o facto de o modelo ser bastante sensível ao ruído, uma vez que o sinal, apesar de não apresentar picos significativos, apresenta ruído, tornando o modelo inadequado, na forma atual, para aplicação em contexto industrial.

De seguida são apresentados os resultados obtidos, em formato de gráfico, da deteção de anomalias de cada modelo.

Nos gráficos estão representados os MSE de cada modelo, à exceção de COPOD, DAGMM, ECOD e HBOS, que utilizam um *score* próprio em vez de MSE.

A representação dos gráficos dos modelos é coincidente, ao nível do tempo, com o sinal original, daí os picos observados nos gráficos e assinalados como anomalias serem coincidentes com os picos no sinal original.

No primeiro *dataset*, praticamente todos os modelos geraram gráficos idênticos, assinalando anomalias nas mesmas zonas e praticamente na mesma quantidade. A maior diferença encontra-se no Transformer Autoencoder e no VAE. Estes dois modelos apresentam um comportamento bastante característico em que é possível

observar que o MSE ao longo do sinal era praticamente nulo e as ocasiões em que houve picos no sinal original ficaram bem assinaladas com picos bastante evidentes no gráfico destes modelos. Autoencoder, TCN Autoencoder e DAGMM também apresentam um comportamento semelhante. Quando os modelos geram gráficos com MSE, ou *score*, praticamente nulo significa que a saída representada pelo modelo apresenta muito poucas diferenças em relação à entrada.

No segundo *dataset* os modelos já assinalam um maior número de anomalias, o que pode fazer sentido após uma breve análise visual do gráfico dos dados de vibração em bruto. O comportamento apresentado pelos modelos é, em geral, semelhante ao comportamento apresentado no primeiro *dataset*, sendo que, em ambos, constam picos isolados no sinal.

No terceiro *dataset* já se observam gráficos bastante distintos em relação aos anteriores, todos os modelos apresentaram um comportamento oscilatório na deteção e, inclusive, deteções em zonas completamente distintas do *dataset*.

De frisar que o equipamento não apresentava nenhuma anomalia à data da recolha dos dados, os picos devem-se a interações entre a ferramenta e a pedra.

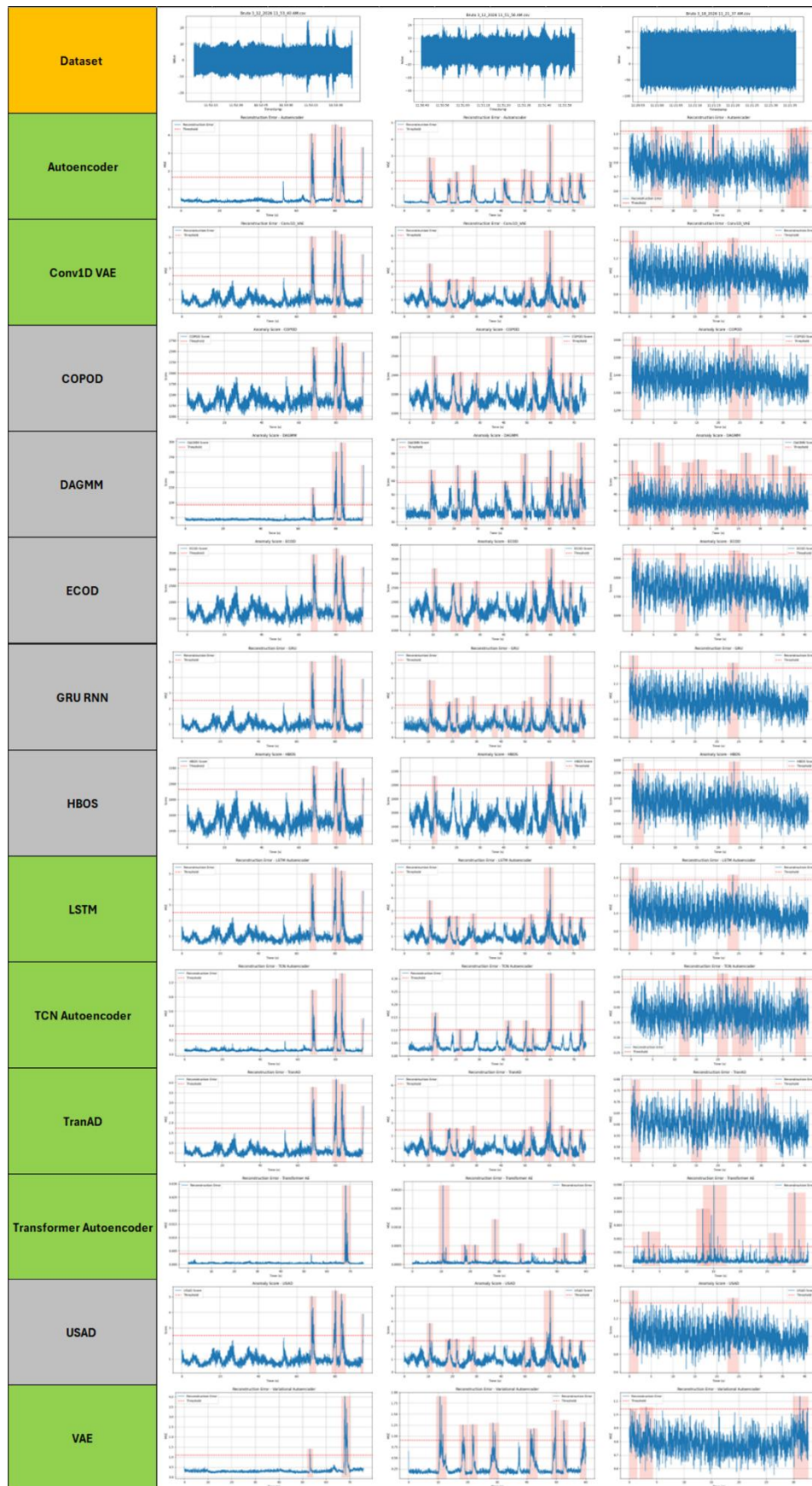


Figura 36 – Resultados Detecção de Anomalias

5. Conclusão

O desenvolvimento deste trabalho permitiu demonstrar a aplicabilidade de técnicas de análise de dados e inteligência artificial na implementação de sistemas de manutenção preventiva baseada na condição dos equipamentos em ambientes industriais. Através da análise de sinais de vibração foi possível explorar diferentes métodos de detecção de anomalias capazes de identificar desvios relativamente ao comportamento normal do equipamento.

A utilização de modelos de aprendizagem não supervisionada mostrou-se particularmente adequada para este cenário, uma vez que, em aplicações industriais reais, a disponibilidade de dados devidamente classificados como falhas é frequentemente limitada. As diferentes arquiteturas de *Deep Learning* avaliadas permitiram estudar abordagens com diferentes capacidades de extração de características, desde modelos baseados em reconstrução através de *Autoencoders* até arquiteturas mais complexas orientadas para dependências temporais.

A comparação entre modelos evidenciou que a escolha da arquitetura tem um impacto significativo no desempenho da detecção, principalmente devido às características dos sinais analisados, onde existem padrões temporais e variações associadas ao funcionamento dos equipamentos. Modelos com capacidade de aprendizagem de relações temporais e extração automática de características apresentam vantagens em cenários onde a informação relevante está distribuída ao longo do sinal.

A integração destes modelos numa arquitetura baseada em *Edge Computing* reforça a viabilidade da solução em aplicações industriais, permitindo realizar processamento local, reduzir a dependência de sistemas externos e obter respostas mais rápidas. Desta forma, o sistema desenvolvido aproxima-se de uma solução aplicável em contexto real, capaz de apoiar estratégias de manutenção baseadas no estado atual dos equipamentos.

Como resultado, o projeto permitiu desenvolver uma solução integrada para monitorização do estado de equipamentos industriais, combinando aquisição de dados, processamento, análise inteligente e detecção automática de comportamentos anómalos. A implementação realizada demonstra como a aplicação conjunta de técnicas de *Machine Learning*, *Deep Learning* e *Edge Computing* pode contribuir para sistemas industriais mais inteligentes, permitindo transformar dados provenientes dos equipamentos em informação relevante para suporte à manutenção.

A abordagem desenvolvida evidencia o potencial destas tecnologias na transição para modelos de manutenção baseados na condição real dos equipamentos, reduzindo a dependência de intervenções preventivas baseadas apenas em períodos fixos de operação. Desta forma, este trabalho contribui para a aplicação prática dos conceitos associados à Indústria 4.0, aproximando as técnicas de inteligência artificial de cenários industriais reais.

6. Anexos

**MODBUS TEST CONNECTION
PROCEDURE**

Modpoll - Master (<https://www.modbusdriver.com/modpoll.html>)
 Diagslave - Slave (<https://www.modbusdriver.com/diagslave.html>)

1º Passo - Entrar na pasta diagslave
 cd Desktop/diagslave/win

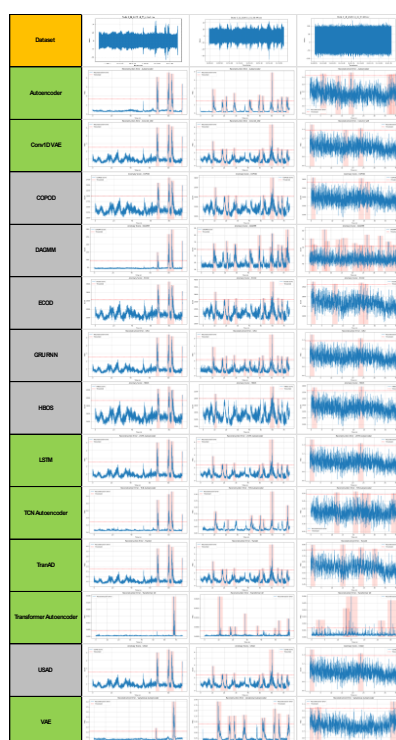
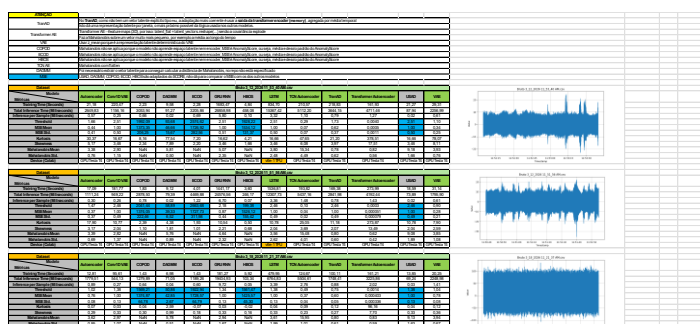
2º Passo - Mostrar opções diagslave
 diagslave -h

3º Passo - Iniciar diagslave com protocolo TCP na porta 502
 Diagslave -m tcp -p 502

4º Passo - Abre novo terminal e entrar na pasta modpoll
 cd Desktop/modpoll/win

5º Passo - Mostrar opções modpoll
 modpoll -h

6º Passo - Arrancar com modpoll
 modpoll -m tcp 127.0.0.1



Referências

- [1] Hossain, M. I., Sanim, S., Kamruzzaman, S. M., Yesmin, M., Sayem, M. S., & Shufian, A. (2025). Anomaly detection in industrial machinery using machine learning and deep learning techniques with vibration data for predictive maintenance. In *Proceedings of the 2nd International Conference on Advanced Innovations in Smart Cities (ICAISC)* (pp. 1–6). IEEE.
- [2] Jablonski, A. (2025). Automatic anomaly detection in large vibration datasets. In *Proceedings of the International Conference on Control, Automation and Diagnosis (ICCAD)* (pp. 1–6). IEEE.
- [3] Fath, N. M. N., Shiddiqi, A. M., Mogi, I. K. A., Rizki, A. M., Syaefudin, M. A., & Rahma, F. (2025). Detecting computer network attacks in time series network data using transformer-based models. In *Proceedings of the 15th International Conference on Information & Communication Technology and System (ICTS)* (pp. 1–6). IEEE.
- [4] Ghorbani, R., Reinders, M. J. T., & Tax, D. M. J. (2024). RESTAD: Reconstruction and similarity based transformer for time series anomaly detection. In *Proceedings of the IEEE 34th International Workshop on Machine Learning for Signal Processing (MLSP)* (pp. 1–6). IEEE.
- [5] Cheng, W., Li, Z., & Cheng, F. (2022). Deep robust autoencoder based framework for bearing fault detection. In *Proceedings of the Global Reliability and Prognostics and Health Management Conference (PHM-Yantai)* (pp. 1–6). IEEE.
- [6] Doorsamy, W., & Bokoro, P. (2024). An investigation into unsupervised anomaly detection for data-driven predictive maintenance. In *Proceedings of the IEEE 12th International Conference on Intelligent Systems (IS)* (pp. 1–12). IEEE.
- [7] Lakey, D., & Schlippe, T. (2024). A comparison of deep learning architectures for spacecraft anomaly detection. In *Proceedings of the IEEE Aerospace Conference* (pp. 1–11). IEEE.
- [8] Malviya, V., Mukherjee, I., & Tallur, S. (2022). Edge-compatible convolutional autoencoder implemented on FPGA for anomaly detection in vibration condition-based monitoring. *IEEE Sensors Letters*, 6(4), 1–4.
- [9] Styaniadis, A., Loufakis, M., Symeonidis, P., Ioannidis, D., Tzovaras, D., & Kourmpetis, I. (2024). Vibration-based anomaly detection on weather radar rotating machinery using one-dimensional convolutional autoencoder. In *Proceedings of the International Conference on Intelligent Perception and Computer Vision (CIPCV)* (pp. 38–45). IEEE.
- [10] Kim, K., Gwak, M., Ryu, S. H., & Park, P. (2025). Reconstruction-based anomaly detection in vibration data with deep state space models. In *Proceedings of the Chinese Control Conference (CCC)* (pp. 6987–6991).
- [11] Haraguchi, H., & Shimizu, T. (2025). Anomaly detection in CNC machining using a dynamic-input LSTM model. In *Proceedings of the 13th International Conference on Control, Mechatronics and Automation (ICCMA)* (pp. 388–393). IEEE.
- [12] Li, H., Kong, X., Shen, G., Yan, X., Yang, Y., & Collotta, M. (2024). Point-correlate adversarial transformer for unsupervised multivariate time series anomaly detection. In *Proceedings of the 27th International Conference on Computer Supported Cooperative Work in Design (CSCWD)* (pp. 297–302). IEEE.
- [13] Wang, Z., Qian, D., Xiao, Z., & Zhang, X. (2024). An anomaly detection method based on continuous wavelet transform and deep convolutional generative adversarial network that only requires normal sample training. In *Proceedings of the 8th International Conference on Electrical, Mechanical and Computer Engineering (ICEMCE)* (pp. 348–353). IEEE.

- [14] Zhu, N., Dong, T. X., & Peng, D. (2025). *Adversarial frequency component reconstruction constraint for helicopter vibration signal anomaly detection: An unsupervised dual-domain approach*. *IEEE Transactions on Instrumentation and Measurement*, 74, 1–11.
- [15] Zheng, L., et al. (2019). *Research on integrated SVDD rotating machinery vibration fault detection method based on deep autoencoder*. In *Proceedings of the IEEE International Conference on Energy Internet (ICEI)* (pp. 531–536). IEEE.
- [16] Mehrabi, A., Dennler, N., Bethi, Y., van Schaik, A., & Afshar, S. (2024). *Real-time anomaly detection using hardware-based unsupervised spiking neural network (TinySNN)*. In *Proceedings of the IEEE International Symposium on Industrial Electronics (ISIE)* (pp. 1–8). IEEE.
- [17] Z. Li, Y. Zhao, X. Hu, N. Botta, C. Ionescu and G. H. Chen, "ECOD: Unsupervised Outlier Detection Using Empirical Cumulative Distribution Functions," in *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 12, pp. 12181-12193, 1 Dec. 2023
- [18] Audibert, Julien. (2021). *Unsupervised anomaly detection in time-series*.
- [19] Thill, M., Konen, W., Bäck, T. (2020). *Time Series Encodings with Temporal Convolutional Networks*. In: Filipič, B., Minisci, E., Vasile, M. (eds) *Bioinspired Optimization Methods and Their Applications*. BIOMA 2020. *Lecture Notes in Computer Science* (), vol 12438. Springer, Cham.
- [20] Julien Audibert, Pietro Michiardi, Frédéric Guyard, Sébastien Marti, and Maria A. Zuluaga. 2020. *USAD: UnSupervised Anomaly Detection on Multivariate Time Series*. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '20)*. Association for Computing Machinery, New York, NY, USA, 3395–3404.
- [21] Tuli, S., Casale, G., & Jennings, N. R. (2022). *TranAD: Deep Transformer Networks for Anomaly Detection in Multivariate Time Series Data*. In *Proceedings of the VLDB Endowment* (Vol. 15, No. 6, pp. 1201–1214).