UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO DE JOINVILLE
CURSO DE ENGENHARIA MECATRÔNICA

VICTOR ELÍZIO PIEROZAN

INTELIGÊNCIA ARTIFICIAL APLICADA PARA DETECÇÃO DE *KNOCK* EM MOTORES AUTOMOTIVOS

Joinville
2022

VICTOR ELÍZIO PIEROZAN

INTELIGÊNCIA ARTIFICIAL APLICADA PARA DETECÇÃO DE *KNOCK* EM MOTORES AUTOMOTIVOS

Trabalho de Conclusão de Curso apresentado como requisito parcial para obtenção do título de bacharel em Engenharia Mecatrônica no curso de Engenharia Mecatrônica , da Universidade Federal de Santa Catarina, Centro Tecnológico de Joinville.

Orientador: Dr. Giovani Gracioli

Joinville
2022

AGRADECIMENTOS

À minha família e amigos pelo suporte e companheirismo ao longo dessa jornada.

Aos todos os meu professores pelo valiosos ensinamentos.

Aos membros do Laboratório de Integração de Software e Hardware pelo trabalho conjunto.

Ao time da Renault do Brasil pela cooperação.

À Fundação de Desenvolvimento da Pesquisa pelo patrocínio do projeto *Intelligent Acquisition and Analysis System for ECUs*.

À toda comunidade científica e de código aberto pela democratização do conhecimento.

Maybe a 3090 is already a human brain. (George Hotz, 2022)

RESUMO

No uso de técnicas de aquisição de informações sobre um sistema, uma aplicação essencial dos dados é a detecção de falhas. Diversos métodos existem para realizar tal tarefa, porém, muitos exigem conhecimento específico de domínio e modelos complexos do sistema avaliado. O advento dos algoritmos de aprendizado de máquina abriu a possibilidade de, automaticamente, se criar modelos do sistema a partir dos dados produzidos, no entanto, ainda há necessidade de se escolher o melhor algoritmo e otimizar os hiper-parâmetros para um dado problema. Assim, este trabalho implementa uma abordagem de detecção de falhas a partir de dados em série temporal, de modo a fornecer uma visão geral das técnicas existem e como aplicá-las. Para isso, o problema abordado foi *Knock* em motores automotivos, uma falha de combustão que gera ruído e desgaste. Onde os dados são coletados pela Electronic Control Unit (ECU) gerando a série temporal de variáveis do sistema, além disso, a própria ECU também tem um algoritmo de detecção de falhas otimizado manualmente por especialistas, o que possibilita a avaliação dos métodos aqui apresentados. Por fim, diversas métricas de desempenho são coletadas e comparadas para ilustrar as diferenças e se decidir a melhor abordagem para o problema em questão.

Palavras-chave: Inteligência Artificial. Série Temporal. Motor. Knock.

ABSTRACT

In the use of techniques for acquiring information about a system, an essential application of the data is fault detection. Several methods exist to accomplish this task, however, many require specific domain knowledge and complex models of the evaluated system. The advent of machine learning algorithms opened the possibility of automatically creating models of the system from the data produced, however, there is still a need to choose the best algorithm and optimize the hyper-parameters for a given problem. Thus, this work implements a fault detection approach from time series data, in order to provide an overview of the techniques that exist and how to apply them. For this, the problem addressed was *Knock* in automotive engines, a combustion failure that generates noise and wear, where the data is collected by the Electronic Control Unit (ECU) generating the time series of system variables, in addition, the ECU itself also has a fault detection algorithm optimized by hand by experts, which makes it possible to evaluate the methods presented here. Finally, various performance metrics are collected and compared to illustrate the differences and decide the best approach for the problem at hand.

Keywords: Artificial Intelligence. Time Series. Engine. Knock.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

IA	Inteligência Artificial
AM	Aprendizado de Máquina
CI	Combustão Interna
LISHA	Laboratório de Integração Software Hardware
ECU	*Electronic Control Unit*
IASE	*Intelligent Acquisition and Analysis System for ECUs*
CTJ	Centro Tecnológico de Joinville
UFSC	Universidade Federal de Santa Catarina
Fundep	Fundação de Desenvolvimento da Pesquisa
IoT	Internet das Coisas
RNA	Rede Neural Artificial
RNC	Rede Neural Convolucional
VP	Verdadeiro Positivo
VN	Verdadeiro Negativo
FP	Falso Positivo
FN	Falso Negativo

LISTA DE SÍMBOLOS

X	Tensor de entrada
Y	Tensor alvo
$\hat{Y}$	Tensor estimado (saída)
X_{test}	Tensor de entrada do conjunto de teste
$\hat{Y}_{train}$	Tensor estimado do conjunto de treino
$\|D\|$	Tamanho do conjunto D ou da primeira dimensão do tensor D
P_{ar}	Pressão de ar do coletor
T_{ar}	Temperatura de ar do coletor
N_m	Velocidade do motor
V_{ar}	Carga de ar do motor
S_i	Vibração (*Knock Noise*) no cilindro i
S_{avg}	Vibração (*Knock Noise*) média
K_i	Contador de detecção de Knock no cilindro i

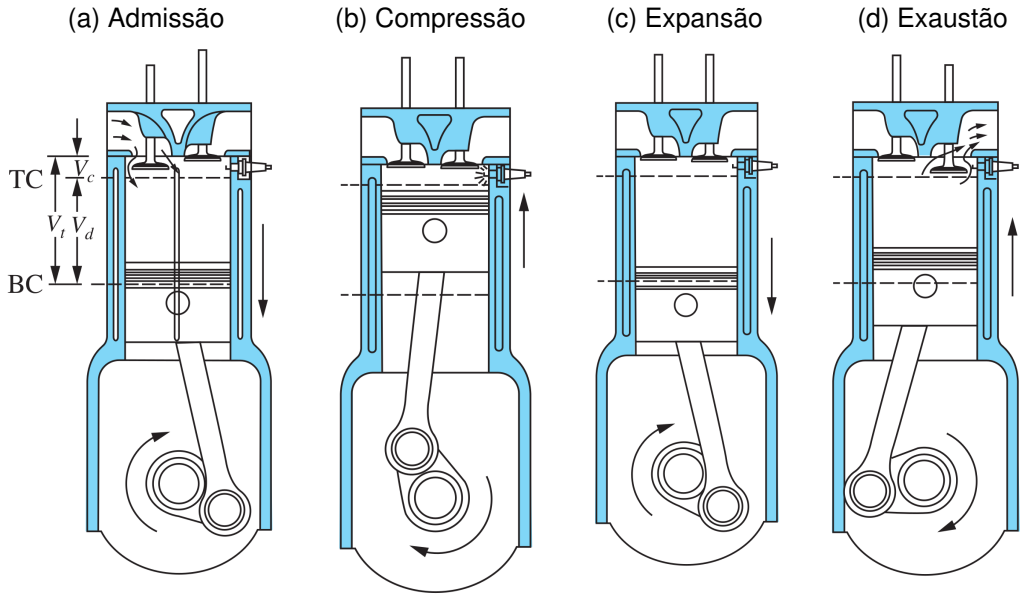
SUMÁRIO

1 INTRODUÇÃO

Este trabalho foi realizado no contexto do projeto *Intelligent Acquisition and Analysis System for ECUs* (IASE), um esforço conjunto do Laboratório de Integração Software Hardware (LISHA), do Centro Tecnológico de Joinville (CTJ), da Universidade Federal de Santa Catarina (UFSC) e da Renault, apoiado pelo Programa Rota 2030 da Fundação de Desenvolvimento da Pesquisa (Fundep). O projeto visa investigar a possibilidade de aplicar técnicas de Inteligência Artificial (IA) dentro do paradigma da Internet das Coisas (IoT) com o objetivo otimizar a operação de motores de Combustão Interna (CI), particularmente em respeito à calibração do controlador e detecção de anomalia. O tema desde trabalho explora o uso de IA para a detecção de um tipo específico de anomalia em motores de CI, o *Knock*, também conhecido como batida de pino.

O funcionamento de motores de combustão interna com ignição por faísca pode ser dividida em quatro fases (HEYWOOD, 2018), exemplificadas na Figura 1, onde entre as fases de compressão e expansão ocorre a combustão da mistura ar-combustível, inciada pela faísca de uma descarga elétrica entre os terminais de um componente chamado vela de ignição. O momento no ciclo em que a combustão ocorre é essencial para operação do motor, afetando desempenho e longevidade da máquina.

Figura 1 – Ciclo de quatro tempos

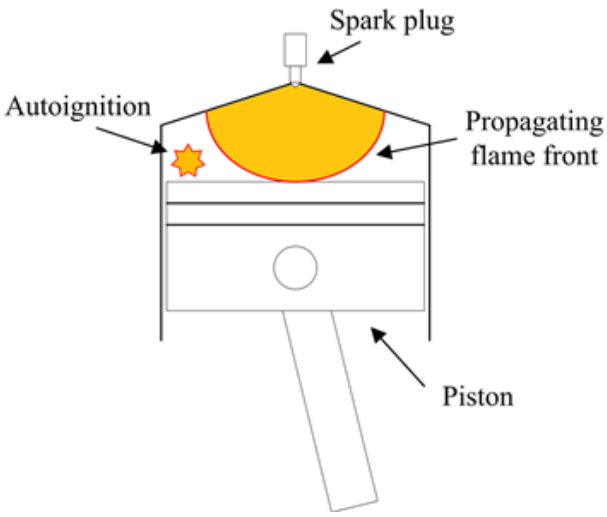


Fonte: Heywood (2018, p. 9) (Adaptado)

O *Knock* é um fenômeno anormal que ocorre em motores à combustão e a literatura (ZHEN et al., 2012) indica que a causa é a combustão espontânea da mistura de ar-combustível antes da frente de chama gerada pela vela de ignição, como pode ser observado na Figura 2, o choque de ambas frentes de chama resulta em um som metálico característico, possíveis danos ao equipamento e redução de desempenho do motor.

Devido às reservas limitadas de combustíveis fósseis e padrões legislativos mais rigorosos de emissões de gases efeito estufa, é de interesse de fabricantes de automóveis, como a Renault, reduzir a ocorrência de *Knock* em seus motores. Nota-se que existe um *trade-off* entre as diversas características do funcionamento de um motor, como potência máxima, eficiência de combustível e ruído. O desejável é chegar-se à um balanço ótimo entre elas e isto é realizado através do controle preciso da operação de todo o sistema.

Figura 2 – Exemplo de Knock



Fonte: Sharma (2018, p. 5)

Existem diversas técnicas analíticas para detectar ocorrência de *Knock* (ZHEN et al., 2012), elas podem ser classificadas em duas amplas categorias: diretas e indiretas. As diretas se baseiam na medição e estudo direto dos parâmetros internos do cilindro, como temperatura e pressão. As indiretas utilizam medidas externas ao cilindro como análise sonora e de vibração do bloco do motor.

Os procedimentos explorados neste trabalho se assemelham aos de análise de vibração do bloco do motor (MILLO; FERRARO, 1998), pois os sinais utilizados são similares, porém possuem a vantagem de não necessitar de calibração fina por um especialista nem de modelagem matemática manual do fenômeno, por se basearem em Aprendizado de Máquina (AM). Estas técnicas permitem o desenvolvimento e

refinamento de algoritmos de detecção a partir dos dados gerados pelos sensores já existentes nos carros, desprezando a necessidade de um especialista de domínio e abrindo portas para expansão da aplicação das técnicas de AM em outras categorias de falha e de outras vias de pesquisa, como predição e mitigação.

O AM é um subcampo da IA de rápido crescimento nos últimos anos e expandiu para aplicações em várias áreas, como visão computacional, controle robótico e reconhecimento de fala, entre outras (JORDAN; MITCHELL, 2015). Parte da atratividade do campo é que as técnicas apresentadas permitem treinar um algoritmo mostrando-o as entradas e saídas desejadas ao invés de programá-lo manualmente prevendo a resposta correta para todas as entradas possíveis.

O objetivo deste trabalho é desenvolver abordagens de AM para detecção de Knock em motores automotivos e realizar uma análise comparativa entre elas, visando explorar a aplicabilidade das mesmas à otimização e calibração de motores CI.

A metodologia do trabalho consiste na utilização de uma base de dados criada a partir da extração dos dados disponíveis pela *Electronic Control Unit* (ECU) de um veículo fornecido pela Renault em experimentos realizados pelo LISHA no CTJ para o desenvolvimento, teste e comparação dos algoritmos aqui apresentados.

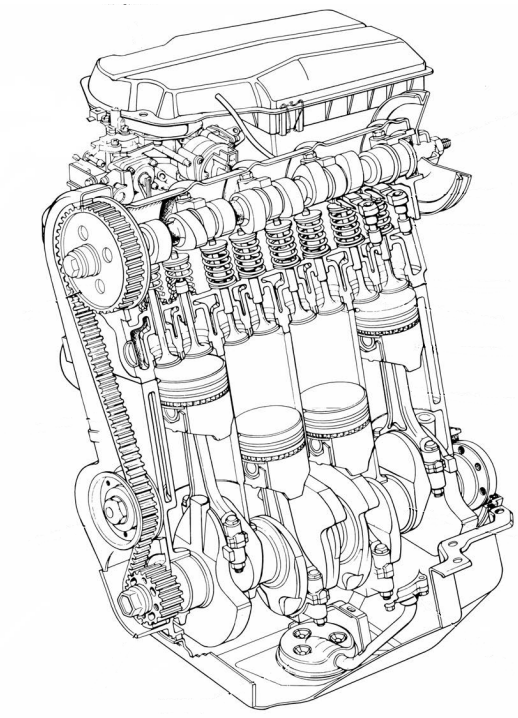
2 FUNDAMENTAÇÃO TEÓRICA

As próximas seções abordam as teorias e técnicas utilizadas neste trabalho. Faz-se necessário discorrer sobre os métodos clássicos de detecção de *Knock* para a compreensão das limitações dos mesmos e do fenômeno em si. Também é preciso dissertar sobre os conceitos básicos de aprendizado de máquina e fundamentos dos algoritmo implementados.

2.1 MOTORES DE COMBUSTÃO INTERNA

O propósito de motores CI é gerar potência mecânica através da energia química contida em um combustível, tal processo ocorre pela combustão de uma mistura combustível-comburente. Nesse caso gasolina e ar atmosférico se misturam, entram em combustão criando um gás que aquece e expande movimentando um pistão. Para os propósitos deste trabalho, são considerados motores CI de quatro cilindros e de quatro tempos de ignição por faísca, exemplificado na Figura 3, também chamados de motores Otto, em homenagem ao seu inventor Nicolaus Otto (HEYWOOD, 2018).

Figura 3 – Motor Otto de quatro cilindros



Fonte: Heywood (2018, p. 12)

As fases do motor Otto representam quatro movimentos lineares do pistão, e duas revoluções do virabrequim, executando em um ciclo, composto por:

1. Admissão: O pistão começa no topo do cilindro e desce, expandindo o volume da câmara de combustão, permitindo a entrada da mistura ar-combustível pela válvula de admissão.
2. Compressão: O pistão sobe novamente, comprimindo a mistura. Ao final dessa fase, ocorre a combustão, rapidamente aumentando a pressão na câmara de combustão.
3. Expansão: Os gases produzidos na combustão expandem forçando o pistão para baixo, girando o virabrequim. Aproximadamente cinco vezes mais trabalho é realizado sobre o pistão do que ele realiza durante a compressão. Ao fim desta fase, a válvula de exaustão abre, liberando a pressão.
4. Exaustão: O pistão sobe e a válvula de exaustão abre, permitindo a saída dos gases remanescentes. Ao fim desta fase a válvula de exaustão fecha e a válvula de admissão abre, reiniciando o ciclo.

Vale notar que a sincronização e controle de todas partes móveis do motor tem efeito significativo sobre a operação do motor, por exemplo, o ângulo do virabrequim, consequentemente a altura do pistão, na fase de compressão em que a combustão é iniciada é chamado de avanço. Aumentar o avanço, aumenta a taxa de compressão da mistura ar-combustível, podendo aumentar torque, porém com esse aumento de pressão há a probabilidade da ocorrência de uma anomalia chamada de *Knock*.

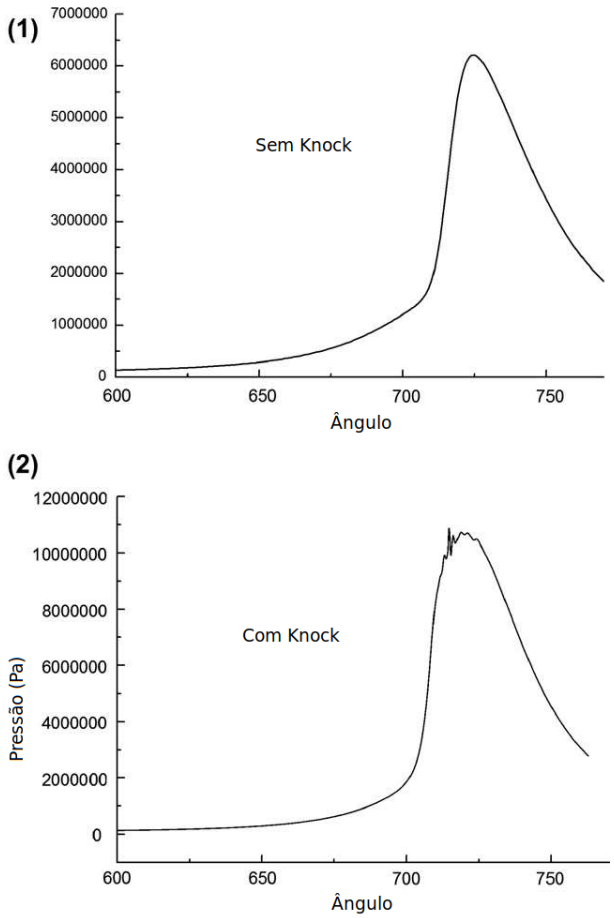
2.2 KNOCK

O *Knock* é um fenômeno bem conhecido e é um empecilho para os aprimoramentos de motores de CI, se persistir por longos períodos de tempo pode causar efeitos como (ZHEN et al., 2012):

- Quebra de anéis de pistão;
- Erosão da cabeça do cilindro;
- Derretimento do pistão;
- Limita taxa de compressão do motor ou desempenho de aceleração do veículo;
- Aumento na poluição do ar;
- Redução na eficiência do motor;
- Danos estruturais ao motor em longo prazo;
- Ruídos indesejados.

Por mais de um século pesquisadores tentam entender as causas do *Knock*, existem duas teorias principais, Detonação e Auto-Ignição. A teoria de Detonação assume que é a própria frente de chama gerada pela vela que reflete entre as paredes do cilindro e que gera o *Knock*.

Figura 4 – Curvas de pressão sem e com *Knock*



Fonte: Zhen et al. (2012) (Adaptado)

A teoria mais aceita é a de Auto-Ignição, onde após a ignição pela faísca da vela, o gás que ainda não sofreu combustão é comprimido e aquecido pelo gás já combustado, atingindo seu ponto de auto-ignição, resultando em combustão espontânea em um ou mais pontos diferentes da vela. A oscilação das diferentes ondas de choque na câmara de combustão resulta no som metálico característico da batida de pino. Ao se analisar as curvas de pressão dentro do cilindro, como na Figura 4 é possível visualizar as vibrações de alta frequência causadas pelo *Knock*.

Descrever o fenômeno de *Knock* é complexo e se faz necessário compreender as reações ocorrendo dentro da câmara de combustão. Ao longo das décadas foram desenvolvidos métodos experimentais e de modelagem para detecção e previsão de *Knock*, e estes são discutidos no Capítulo 3, mas o foco deste trabalho são técnicas baseadas no AM.

2.3 APRENDIZADO DE MÁQUINA

O Aprendizado de Máquina, subcampo da IA, é uma técnica para realizar tarefas focando em derivar um programa de dados coletados sobre o problema ao invés de manualmente escrever a lógica.

> "Diz-se que um programa de computador aprende com experiência E em relação a alguma classe de tarefas T e medida de desempenho P, se seu desempenho em tarefas em T, medido por P, melhora com experiência E." (MITCHELL, 1997, p. 2, tradução nossa)

2.3.1 Tarefa

Existem várias classes canônicas de tarefa no contexto de aprendizado de máquina, entre elas (SHARMA; KUMAR, 2017):

- Classificação;
- Regressão;
- Reconhecimento de fala;
- Reconhecimento de imagem;
- Detecção de anomalia.

Para se atingir o objetivo geral deste trabalho, será feito uso da detecção de anomalia e classificação. É importante ressaltar que a tarefa de detecção de anomalia pode ser enxergada como uma tarefa de classificação de duas classes: anômalo e saudável. Contudo, para realizar a tarefa um algoritmo de aprendizado de máquina precisa de experiência.

2.3.2 Experiência

Os algoritmos variam de acordo com o tipo de experiência, ou dados de treinamento, e podem ser classificados nas seguintes categorias (GÉRON, 2019):

- Supervisionado: cada conjunto de entrada para os dados possui um conjunto alvo de saída.
- Não-Supervisionado: somente os dados de entrada e não há saída correta nos dados de treinamento.
- Semi-Supervisionado: entre Supervisionado e Não-Supervisionado, ou com muitos dados do conjunto de saída faltando.
- Reforço: o algoritmo aprende tomando ações e recebendo respostas de um ambiente.

Neste trabalho são abordadas técnicas de aprendizado Supervisionado e Não-Supervisionado utilizando uma arquitetura de Rede Neural Artificial (RNA).

2.3.3 Desempenho

A medida de desempenho de uma RNA é obtida através da Função de Custo J, onde as entradas podem variar de acordo com o modelo mas a saída é sempre um único número. Durante o treinamento o otimizador é o algoritmo responsável por atualizar os parâmetros θ da RNA para minimizar o resultado de $J(\theta)$. A classe de otimizador mais popular é a Descida por Gradiente (GÉRON, 2019), onde ela calcula o gradiente de J com respeito aos parâmetros $\nabla_\theta J(\theta)$ e atualiza θ na direção oposta de $\nabla_\theta J(\theta)$ com uma magnitude η, também chamada de Taxa de Aprendizado:

$$\theta_{i+1} = \theta_i - \eta \nabla_\theta J(\theta)$$

Iterativamente se aproximando do mínimo local de $J(\theta)$. Neste trabalho é utilizado o otimizador ADAM (RUDER, 2016), uma variação de Descida por Gradiente.

2.4 TENSORES E NOTAÇÃO

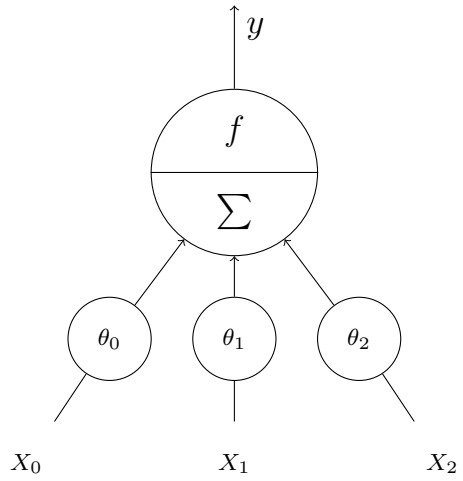
Um conceito necessário para a compreensão de certos algoritmos IA é o Tensor, uma generalização das noções de escalar, vetor e matriz. Por exemplo, um vetor é um tensor de ordem 1 e uma matriz é um tensor de ordem 2. No campo da ciência de computação ele pode ser entendido como um vetor N-dimensional ou ordem N. Para as próximas seções é útil introduzir um pouco de notação:

- X : Tensor de entrada;
- Y : Tensor alvo;
- $\hat{Y}$: Tensor estimados (saída);
- X_{test} : Tensor de entrada do conjunto de teste;
- $\hat{Y}_{train}$: Tensor estimados do conjunto de treino;
- $\|D\|$: Tamanho do conjunto D ou da primeira dimensão do tensor D.

2.5 REDES NEURAIS ARTIFICIAIS

As RNAs, introduzidas pela primeira vez por McCulloch e Pitts em 1943 (MCCULLOCH; PITTS, 1943), são um modelo computacional de como cérebros animais podem executar funções complexas. O elemento básico das RNAs é o Neurônio Artificial, uma unidade simples de computação com múltiplas entradas e uma saída. Os Neurônios Artificiais utilizados nas diversas arquiteturas de RNAs podem variar, porém a ideia central se mantém, se assimilando à Unidade de Limite Linear, exemplificada na Figura 5 introduzida por Frank Rosenblatt (CORNELL AERONAUTICAL LABORATORY, 1957), onde a soma ponderada das entradas passa por uma função de ativação para produzir a saída.

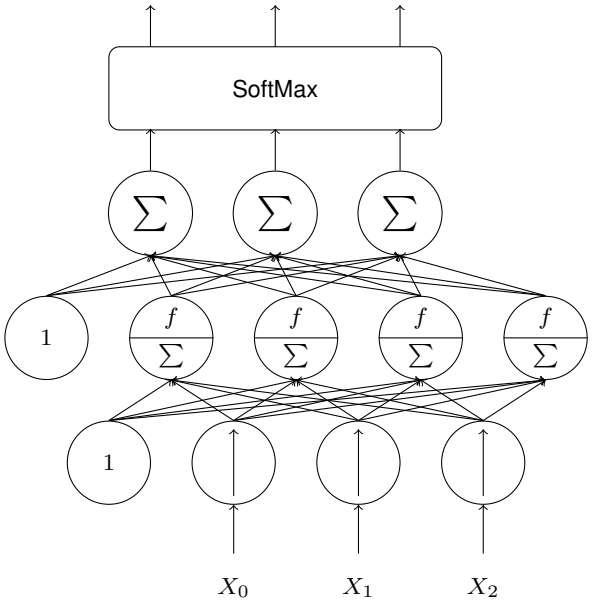
Figura 5 – Unidade de Limite Linear



Fonte: Géron (2019, p. 257) (Adaptado)

Como exemplificado na Figura 6, múltiplos Neurônios Artificiais são colocados em paralelo para formar uma camada e essas camadas são conectadas em série para formar uma RNA.

Figura 6 – Exemplo de uma RNA



Fonte: Géron (2019, p. 261) (Apdaptado)

Assim, na fase de treinamento se ajusta os pesos das conexões, também chamados de parâmetros, de cada Neurônio Artificial com base nos dados de treinamento para que a RNA produza o resultado desejado. Em contraste com os parâmetros da rede, há os chamados hiper-parâmetros, estes são definidos por quem constrói o modelo e não são otimizados pela Descida por Gradiente, ou seja, não são treináveis. Arquiteturas de RNAs recebem diferentes nomes dependendo da disposição

dos Neurônios Artificiais:

2.6 CLASSIFICADOR

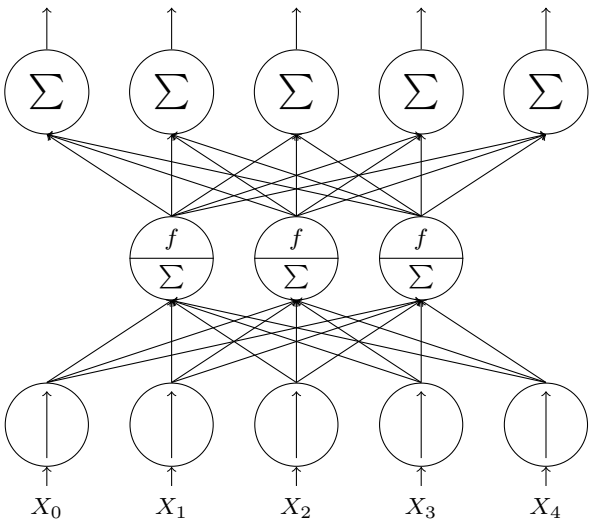
O primeiro método implementado para este trabalho foi um classificador (GÉRON, 2019), classificadores são utilizados para prever variáveis discretas, no caso deste trabalho, prever se uma série temporal de diversas variáveis contém um *Knock* ou não. Usando aprendizado Supervisionado, o algoritmo visa identificar a classe dos dados analisados, ou seja, que o valor predito $\hat{Y}$ seja o valor real Y.

A entrada de um Classificador é um vetor de variáveis X que representa o objeto a ser classificado, e a saída é $\hat{Y}$, que representa a predição do modelo, geralmente representada por um vetor onde em cada posição está a probabilidade associada a cada classe de objeto, para isso se utiliza uma função *SoftMax* para transformar $\hat{Y}$ em um vetor de probabilidades.

2.7 AUTOENCODER

Os AutoEncoders são RNAs capazes de aprender representações compactas dos dados de entrada, a ideia é treinar a rede para reconstruir os dados de entrada, porém forçando os dados a passarem por uma redução de dimensionalidade nas camadas intermediárias, como exemplificado na Figura 7.

Figura 7 – Exemplo de um AutoEncoder



Fonte: Géron (2019, p. 261) (Adaptado)

Essa representação intermediaria pode ser utilizada de diversas maneiras, como modelos geradores de dados, extrator de variáveis para serem utilizadas em outros modelos ou detectores de anomalia. Neste trabalho serão explorados as funcionalidades de extração de variáveis e detecção de anomalia.

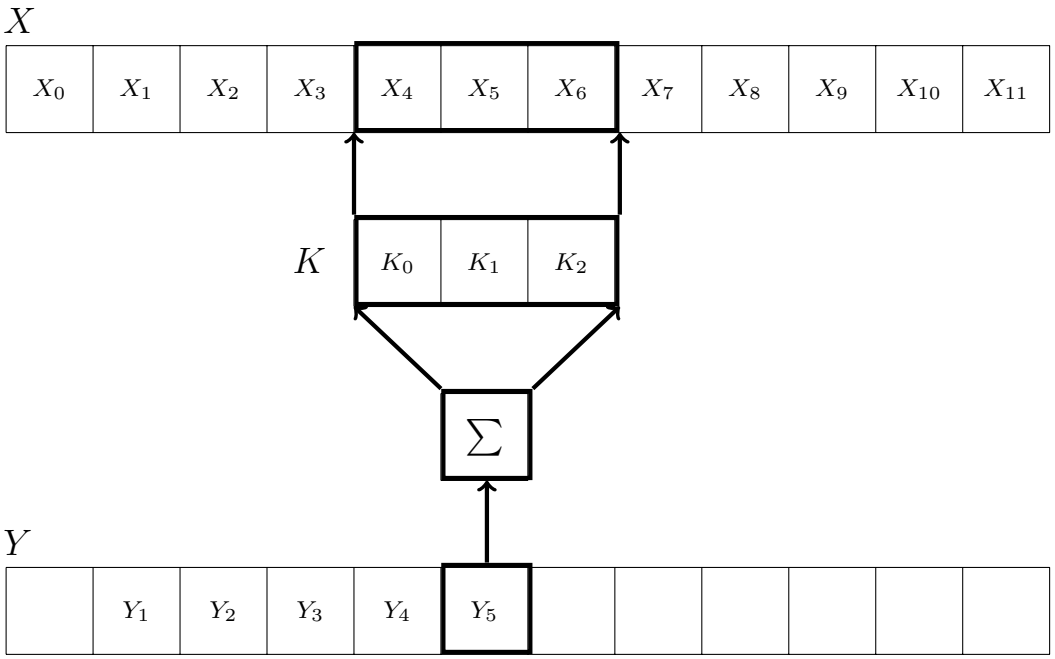
2.8 REDES NEURAIS CONVOLUCIONAIS

As Redes Neurais Convolucionais (RNCs) vêm ganhando popularidade nas últimas décadas (ALBAWI; MOHAMMED; AL-ZAWI, 2017) pela sua habilidade de detectar padrões nos dados, por isso muito utilizadas em classificação de imagens. Neste trabalho deseja-se identificar padrões em sinais temporais multivariados, logo a utilização de convoluções se faz útil.

2.8.1 A Convolução

A operação de convolução consiste de deslizar um tensor K, denominado *kernel*, sobre os dados X ao passo s . Ao se deslizar K sobre X os elementos são multiplicados um a um e somados, resultando em um escalar. Como exemplificado na Figura 8, temos $X_{12\times1}$ e K_{3x1}.

Figura 8 – Convolução 1D



Fonte: Produzida pelo autor.

Contudo existem parâmetros que podem ser ajustados em uma operação de convolução, como o passo em que a janela desliza e se irá preencher os dados que a janela fica pra fora de X. Com este processo, a rede neural otimiza os valores de K para que os padrões mais representativos dos dados sejam extraídos, minimizando a função de custo final.

2.9 ALGORITMOS ESCOLHIDOS

Os algoritmos foram escolhidos de forma a explorar diferentes áreas do AM, são eles:

Tabela 1 – Resumo dos algoritmos propostos

Nome	Categoria	Descrição
Modelo Aleatório	Probabilístico	Escolhe aleatoriamente com base na frequência de *Knock*
Classificador de Sequência	Classificador	Aprende com os dados de entrada e os *Knocks* detectados
AutoEncoder Denso	AutoEncoder	Aprende o padrão do dados de entrada e quando um ponto desvia do padrão aprendido um erro é sinalizado
AutoEncoder Convolucional	AutoEncoder Convolucional	Similar ao AutoEncoder denso, mas aprende em sequências de pontos
Classificador com Extração de variáveis	Classificador Convolucional	Utiliza um Encoder Convolucional para gerar variáveis mais significativas para uma fase de classificação

2.10 TECNOLOGIAS UTILIZADAS

Implementar algoritmos de AM do zero é complexo, por isso foram utilizadas tecnologias existentes no desenvolvimento do modelos aqui apresentados, entre elas[1]:

- Python 3.10.4: linguagem de programação de uso geral e alto nível;
- Numpy 1.22.3: biblioteca de manipulação eficiente de grandes matrizes multi-dimensionais;
- Pandas 1.4.2: biblioteca para análise e manipulação de dados tabulares;
- SkLearn 0.0: biblioteca contendo diversos utilidades e algoritmos de ML;
- TensorFlow 2.9.0: biblioteca de IA e AM, focada no treinamento e inferência RNAs profundas.
- Matplotlib 3.5.2: biblioteca para geração de gráficos.

[1] As bibliotecas na citadas podem ser obtidas através do índice de pacotes python pypi.org

3 TRABALHOS RELACIONADOS

O *Knock* é um fenômeno bem conhecido e existem diversas técnicas de detecção e para a compreensão da aplicabilidade deste trabalho é importante ter uma noção geral delas. Nos tópicos seguintes no intuito de inspiração e comparação são explorados artigos de temas similares, onde estão separados em duas seções: Métodos Analíticos e Métodos de AM.

3.1 MÉTODOS ANALÍTICOS

Estes métodos, aqui denominados analíticos, são as abordagens clássicas que envolvem conhecimento aprofundado sobre a física do fenômeno e podem ser classificados com base na variável do motor utilizada e se esta é interna ou externa à câmara de combustão. Estes são os principais (ZHEN et al., 2012):

- Métodos baseados na análise da pressão interna do cilindro;
- Métodos baseados na análise das vibrações do bloco do motor;
- Métodos baseados na temperatura dos gases de exaustão;
- Métodos baseados em radicais intermediários e análise de espécies;
- Métodos baseados na análise da liberação de calor.

As técnicas diretas, como análise de pressão no cilindro, são mais precisas, porém a medição de valores internos ao cilindro requer sensores caros e instalados dentro do mesmo, os quais entram em contato com misturas quentes e em alta pressão, reduzindo vida útil e acurácia. Por isso muitas vezes são utilizadas como referência para calibração de outros métodos.

As técnicas indiretas, como análise de vibração do bloco do motor, podem utilizar sensores externos ao cilindro, melhorando custo e durabilidade. Porém são altamente dependentes do tamanho e formato da câmara de combustão, os modos ressonantes criados e velocidade do som do meio local, parâmetros os quais podem variar em diferentes pontos de operação do motor.

Os métodos propostos neste trabalho utilizam apenas variáveis indiretas, principalmente as de vibração do motor, pois há a limitação de captar apenas os sinais expostos pela ECU, além de que veículos comerciais geralmente não possuem sensores internos à câmara de combustão.

Dentre métodos analíticos baseados na análise das vibrações do bloco do motor se encontram (MILLO; FERRARO, 1998):

- Valor de pico a pico máximo da pressão ou sinal de vibração filtrada por passa-banda (PPMax);

- Valor quadrado médio da pressão ou sinal de vibração filtrado por passa-banda (MSV);
- Integral do valor absoluto da primeira derivada da pressão ou vibração filtrada por passa-banda (ID).

Porém, todos necessitam de altas frequências de amostragem, na grandeza de Kilohertz, no entanto para a aplicação deste trabalho existe uma limitação de um período de amostragem mínimo de 4 milissegundos, ou seja, 250 Hertz. As razões para essa limitação serão discutidas no capítulo 4. Assim, outra abordagem, como AM, se faz necessária.

3.2 MÉTODOS DE APRENDIZADO DE MÁQUINA

Os métodos de AM são o foco deste trabalho e apresentam a vantagem de dispensar conhecimento aprofundado do fenômeno, os algoritmos inferem a modelagem a partir dos dados coletados. Além disso, podem superar as limitações apresentadas pela aquisição dos dados, que impossibilitam a aplicação de métodos analíticos.

Similarmente à este trabalho, (AL-ZEYADI et al., 2020) fizeram uso de AM com um modelo de rede neural profunda para previsão de falhas em veículos. A rede é treinada com trilhas de auditoria de reparos de veículos. A arquitetura de rede usa as camadas Dense, ReLu, Dropout e SoftMax, com o otimizador ADAM, enquanto a entrada é um vetor de sintomas do veículo (reportado pelo proprietário) e um vetor de características do veículo (medido por um técnico) , e para a saída, um vetor de tipos de falhas.

A solução foi testada com 10 modelos de veículos e comparada com outros algoritmos, como *Deep Convolution 1D* e *Radial Basis Function Support Vector Machine*, teve o melhor desempenho em 6 de 10 modelos com uma precisão de 60%-80%, mas os outros algoritmos eram bastante competitivos.

Sua solução é muito focada em uma ampla variedade de veículos e ajuda os técnicos a encontrar falhas e reparar veículos, mas fornece informações úteis sobre a capacidade de generalização de arquiteturas de redes neurais profundas.

Este trabalho utiliza os mesmos conceitos do artigo, as mesmas camadas e mesmo otimizador, porém o problema é diferente, aqui são utilizados dados temporais obtidos dos sensores do carro em movimento para fazer a detecção de falhas do carro, ao invés de características não temporais obtidas quando o carro vai para reparo.

No conceito de detecção de anomalias em série temporal, (YAN, 2020) abordam o problema de detectar anomalias em motores de turbinas a gás a partir de dados brutos de sensores em duas partes: Descoberta de variáveis e Detecção de anomalia. A solução de descoberta de variáveis proposta é um algoritmo *Stacked Denoising*

Autoencoder (SDAE) treinado apenas com os dados saudáveis e a solução de detecção de anomalia proposta é uma *One-Class Extreme Learning Machine* (ocELM).

Esta técnica serve de inspiração para este trabalho, apesar do problema abordado ser levemente diferente, aqui são motores CI automotivos ao invés de turbinas à gás. Porém, a ideia de usar uma fase de extração de variáveis e uma fase de classificação serviu de inspiração para este trabalho e mostrou bons resultados.

Corroborando esta abordagem, (SHAHID; KO; KWON, 2022) desenvolvem uma técnica de detecção e classificação em tempo real de anomalias em motores à diesel, que defende que as variáveis extraídas na fase de extração oferecem melhor desempenho total na detecção de anomalia do que se elas fossem derivadas manualmente com base no entendimento do fenômeno. Porém o problema abordado é de motores à diesel e a natureza dos dados é diferente, os dados utilizados no artigo tem um período de aquisição de um minuto e o conjunto inteiro se estende por um ano com apenas dez falhas.

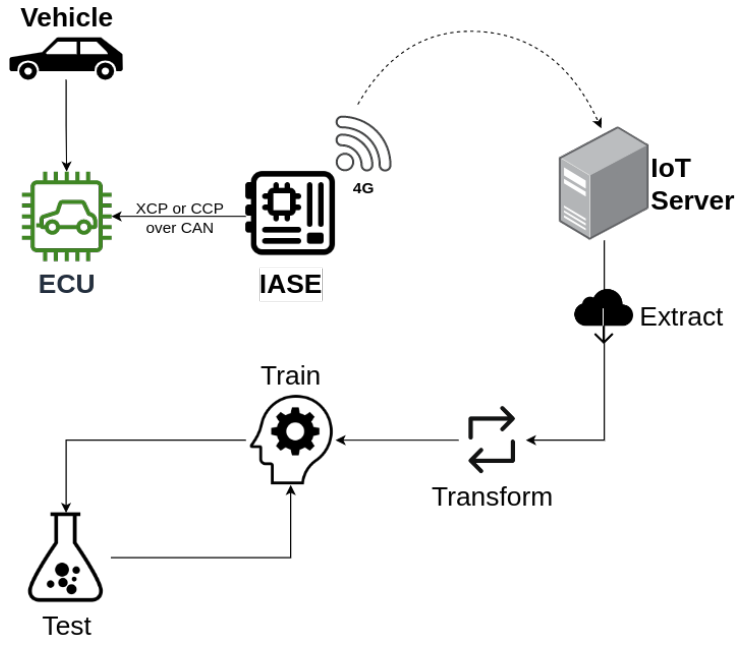
Assim, neste contexo, se explora soluções para detecção de *Knock* em motores automotivos utilizando dados da ECU com base em técnicas de AM.

4 DETECÇÃO DE *KNOCK* COM INTELIGÊNCIA ARTIFICIAL

Neste capítulo são apresentados os materiais e métodos utilizados para o desenvolvimento e treinamento dos algoritmos neste trabalho. Como mencionado no Capítulo 2, o aprendizado de máquina envolve experiência E, o que geralmente se traduz em dados representativos do fenômeno de interesse.

O conjunto de dados utilizado neste trabalho foi gerado utilizando um carro, um Sandero 2016, fornecido pela Renault, onde foram realizados experimentos dentro do CTJ e do Perini Business Park. Para a leitura dos dados foi utilizado o IASE, um sistema desenvolvido pelo LISHA no contexto do projeto Fundep Rota 2030, que lê os dados do carro e os envia via rede 4G a uma plataforma IoT do onde ficam salvos em um banco de dados. Em seguida, os dados são acessados e processados, alimentando os algoritmos de IA para treinamento e teste, como ilustrado na Figura 9.

Figura 9 – Processo de desenvolvimento



4.1 AQUISIÇÃO DOS DADOS

A placa desenvolvida no projeto se conecta à ECU do carro pelo barramento CAN utilizando o protocolo XCP ou CCP e aquisita os dados. Aqui se encontra a limitação aludida no Capítulo 2, pois o barramento CAN/CCP/XCP possui latência e largura de banda que só é possível aquisitar com um período mínimo de 4 milissegundos

e um número máximo de variáveis por experimento, devido a esta limitação as variáveis a serem aquisitadas são selecionadas previamente ao experimento. Após a aquisição dos dados no carro, eles são convertidos para um formato padrão chamado de *SmartData* (FRÖHLICH, 2018), utilizando unidades do Sistema Internacional de Unidades, e enviados para o servidor, assim ficando disponíveis para consulta.

4.2 SELEÇÃO DE VARIÁVEIS

Das diversas variáveis disponibilizadas pela ECU, somente algumas foram selecionadas para a aquisição, com base no entendimento físico do fenômeno, teoria de Auto-Ignição (ZHEN et al., 2012) e em conhecimento de domínio de especialistas de calibração da Renault, são elas:

- P_{ar} : Pressão de ar do coletor;
- T_{ar} : Temperatura de ar do coletor;
- N_m : Velocidade do motor;
- V_{ar} : Carga de ar do motor;
- S_i : Vibração (*Knock Noise*) no cilindro i;
- S_{avg} : Vibração (*Knock Noise*) média;
- K_i : Contador de detecção de Knock no cilindro i.

Cada variável vem acompanhada de uma data e hora (*timestamp*) além do valor em si, permitindo a ordenação temporal das mesmas em uma tabela, ou uma série temporal multivariada.

4.3 PREPARAÇÃO DOS DADOS

Antes do treinamento e inferência, passos de pré-processamento são aplicados aos dados D, ajudando a eliminar informações irrelevantes do modelo, como unidade de temperatura, melhorando desempenho e generalidade.

Dependendo da variável, ela pode ser amostrada com diferentes períodos, logo todas são reamostradas em software com o maior período oferecido pela ECU, dez milissegundos. Além disso, para as variáveis contínuas, como P_{ar}, é feita uma normalização entre $[0, 1]$ utilizando a seguinte fórmula:

$$P_{ar\_norm} = \frac{P_{ar} - P_{ar\_min}}{P_{ar\_max} - P_{ar\_min}} \tag{1}$$

Para as variáveis de contagem, como K_i, o número inicial é irrelevante, logo é feita uma diferenciação discreta, transformando de um contador para a diferença do contador em t com relação a $t - 1$, da seguinte maneira:

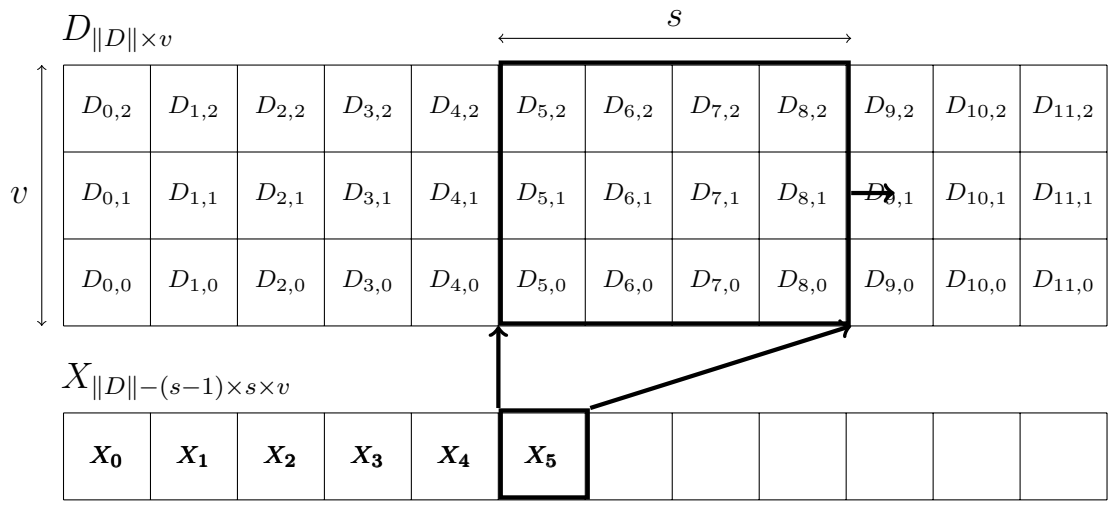
$$K_i'(t) = \begin{cases} 0 & t = 0 \\ K_i(t) - K_i(t-1) & t > 0 \end{cases} \tag{2}$$

Após a diferenciação discreta os valores são transformados em Booleanos, efetivamente resultando em um série temporal da ocorrência do *Knock*. Além disso, nota-se que há uma grande correlação entre a ocorrência de *Knock* entre os cilindros, após a diferenciação discreta, todas as variáveis K_i são fundidas em uma única variável K através de uma soma, resultando na ocorrência de *Knock* em qualquer cilindro naquele instante, sendo assim utilizada como rótulo da presença de falha para os algoritmos de classificação.

4.3.1 Gerando sequências

Para alguns algoritmos não basta um ponto t no tempo como entrada, eles analisam uma sequência dos s últimos pontos aquisitados, permitindo captar anomalias ao longo do tempo ao invés de apenas pontos fora da curva. Esta sequência é representada por uma matriz X tamanho $s \times v$, sendo v o número de variáveis. Cada matriz de entrada X é produzida iterativamente ao deslizar-se uma janela de tamanho s pelos períodos desejados, como ilustrado na Figura 10.

Figura 10 – Geração de sequências por janelamento



Fonte: Produzida pelo autor.

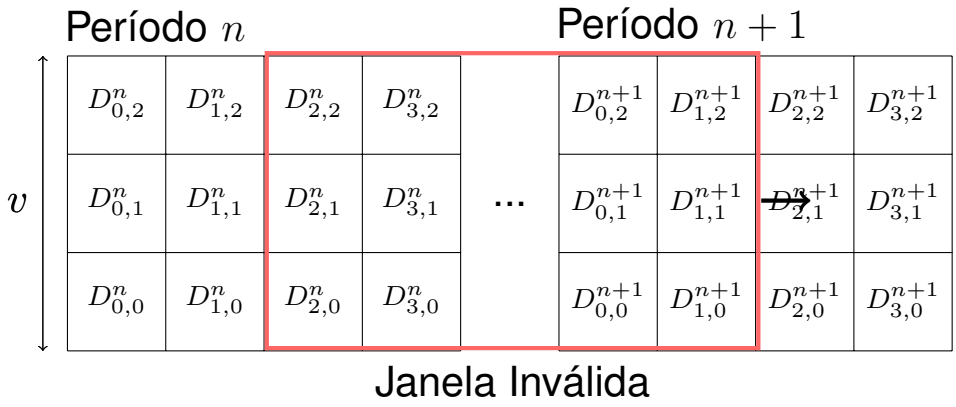
Nota-se que D é um conjunto de dados $\|D\| \times v$, sendo $\|D\|$ o número de pontos de dados aquisitados, ao se gerar as sequências D_{seq} se obtém um tensor $\|D\| - (s-1) \times s \times v$, assim por limitações de memória, D_{seq} é gerado iterativamente ao

longo do treinamento dos algoritmos. Para os algoritmos aqui propostos foi escolhido arbitrariamente $s = 64$, equivalendo a uma janela temporal de 640 milissegundos.

4.3.2 Separação de períodos

Para alguns algoritmos é importante haver uma separação entre dados saudáveis e anômalos, para isso a variável K é utilizada para separar tais períodos. Primeiro todos os intervalos contínuos onde $K \geq 1$ são separados em $[t_0, t_f]$, após isso os intervalos de anômalos são expandidos ao se realizar $[t_0 - g_1, t_f - g_2]$, sendo g_1 e g_2 constantes de tempo, para garantir que apenas dados saudáveis fiquem fora. Depois da expansão pode ser que alguns intervalos estejam sobrepostos, então um algoritmo de simplificação dos intervalos funde a sequência de intervalos em uma sequência onde não há sobreposição. Por fim os dados dentro dos intervalos são rotulados períodos anômalos e fora, saudáveis. No caso dos algoritmos que utilizam D_{seq} o janelamento é aplicado a cada período após a separação, para evitar sequências que abrangem períodos de tempo disjuntos.

Figura 11 – Sequência inválida abrangendo períodos disjuntos



Fonte: Produzida pelo autor.

Como exemplificado na Figura 11, se o janelamento for simplesmente aplicados à concatenação dos períodos de interesse extraídos do conjunto de dados contínuo total, haverão janelas com descontinuidades temporais em D_{seq}, treinando o algoritmo em dados que possivelmente não ocorreriam na realidade.

4.4 DESENVOLVIMENTO DOS MODELOS

Com os dados tratados, os mesmos ficam prontos para serem inseridos na fase de treinamento e teste dos algoritmos. Cada algoritmo necessita de um pré-

processamento específico e tem um mecanismo de funcionamento diferente. Os modelos implementados estão descritos a seguir:

4.4.1 Modelo Aleatório

Para servir de referência foi implementado um modelo aleatório, onde a frequência de anomalias é calculada como:

$$f = \frac{N_{knk}}{N} \tag{3}$$

- f: Frequência de falhas
- N_{knk}: Número de pontos anômalos no conjunto de dados
- N: Número de pontos no conjunto de dados

Assim, o modelo aleatoriamente determina cada ponto como anômalo com uma chance f. Além disso, este modelo é útil para demonstrar a ineficácia da métrica de acurácia neste caso, como discutido no Capítulo 5.

4.4.2 Classificador de Sequência

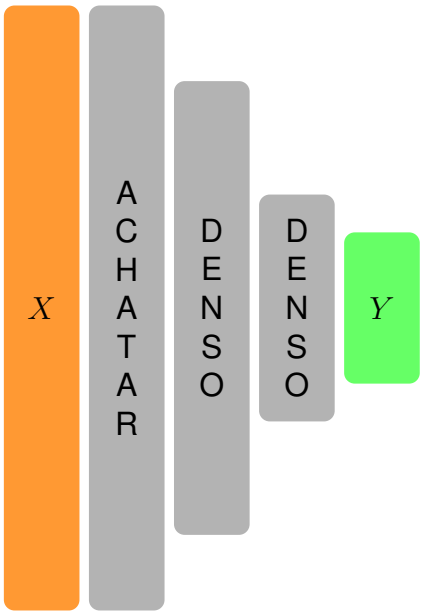
Este modelo é a implementação de um classificador utilizando RNAs, como ilustrado na Figura 12, a primeira camada, simbolizada na figura como *ACHATAR* é uma camada que não altera os dados em si, mas sim seu formato, achatando a matriz de entrada X em um único vetor $s.v \times 1$. As camadas seguintes são camadas densas, assim como na Figura 6, onde a entrada de cada neurônio é a soma ponderada da saída de todos os neurônios da camada anterior. A quantidade de neurônios dessas camadas densas vai diminuindo, ou "afunilando", até uma camada final com apenas um neurônio, o qual possui uma função de ativação sigmoide, resultando na "confiança"do modelo em que a sequência X é anômala. Assim, o modelo é treinado para prever os valores de K como Y_{train}, com a função de custo sendo *BinaryCrossEntropy*.

Para este modelo os conjuntos de treino e teste são simplesmente separados numa proporção 3:1. E para gerar a predição do modelo, valores de saída $\hat{Y}$ maiores que um hiper-parâmetro K são consideradas anomalias e menores que K, consideradas normais.

4.4.3 AutoEncoder Denso

Este modelo segue a arquitetura de um AutoEncoder, similar à Figura 7 com um Codificador e Decodificador formando um "gargalo", no caso deste modelo específico foram utilizadas somente camadas densas, como ilustrado na Figura 13. Além disso, este modelo não usa sequências de pontos, mas sim um único vetor tamanho v com as variáveis naquele instante de tempo como entrada para identificar anomalias.

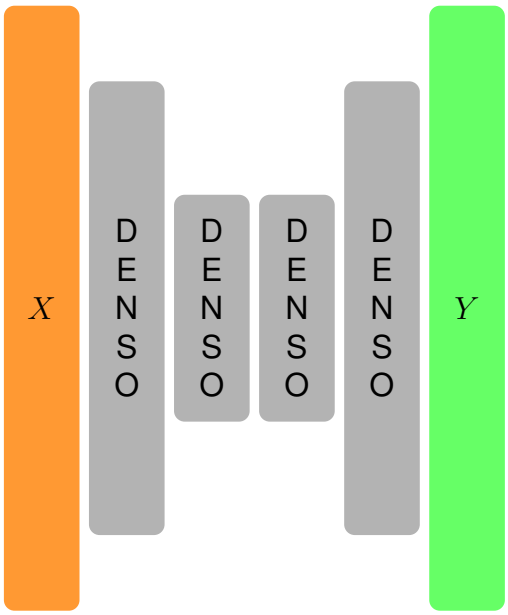
Figura 12 – Arquitetura do Classificador



Fonte: Produzida pelo autor.

Como se deseja aprender um formato mais representativo dos dados, se treina a rede com o Y sendo o próprio X, assim o modelo busca codificar as entradas para a representação latente e que a decodificação seja o mais similar possível as entradas.

Figura 13 – Arquitetura do AutoEncoder Denso



Fonte: Produzida pelo autor.

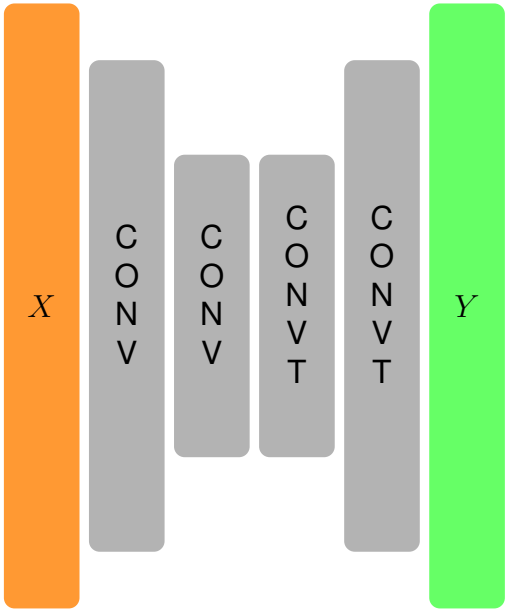
Vale notar que ele é treinado somente com os períodos saudáveis e testado com todos os dados, assim tem-se duas distribuições dos resultados da função de

custo J, J_{train} para os dados de treinamento e J_{test} para os dados de teste. Assim, haverá grande sobreposição entre as duas distribuições, porém na J_{test} espera-se que parte da distribuição esteja concentrada em valores maiores de J, simbolizando os dados que fogem do padrão que o modelo aprendeu a reconstruir. Dessa maneira, define-se um limite K para o qual dados em que $J > K$ são considerados anômalos.

4.4.4 AutoEncoder Convolucional

Este modelo se assemelha muito ao AutoEncoder Denso, a diferença é que este utiliza uma sequência temporal como X, ou seja, uma matriz $s \times v$ das variáveis v ao longo de s amostras. E como observado na Figura 14, ao invés de camadas densamente conectadas, ele utiliza camadas de convolução temporal (1D) para a codificação, representadas por *CONV* na figura e para decodificação, camadas de convolução temporal transposta, representadas por *CONVT* na figura. O passo do deslizamento do *kernel* durante as convoluções é definido como 2, assim gerando a redução de dimensionalidade e compressão da informação.

Figura 14 – Arquitetura do Autoencoder Convolucional



Fonte: Produzida pelo autor.

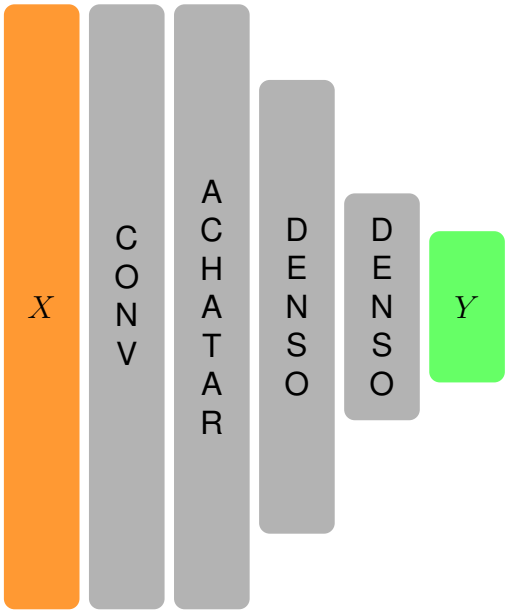
Este algoritmo é treinado da mesma maneira que o AutoEncoder Denso, treinando com os períodos saudáveis, testando com todos e definindo um limite K na função de perda J para a detecção.

4.4.5 Classificador com Extração de variáveis

Este modelo é um classificador, porém ao invés de classificar diretamente os valores nas sequências temporais, primeiro ele extraí uma representação mais

significativa utilizando uma camada de convolução, como ilustrado na Figura 15. Assim, a fase de classificação consegue identificar melhor as diferenças entre as amostras.

Figura 15 – Arquitetura do Classificador com Extração de Variáveis



Fonte: Produzida pelo autor.

Similarmente ao Classificador de Sequências, os conjuntos de sequências são separados numa proporção 3:1 e é definido um hiper-parâmetro K para a determinação das anomalias.

4.5 MÉTRICAS

Ao final do processo de treinamento, o modelo é aplicado sobre os dados de teste X_{test} para se obter o $\hat{Y}_{test}$. Assim as duas séries temporais Booleanas Y_{test} e $\hat{Y}_{test}$ são utilizadas para avaliação, para isso é necessário definir métricas.

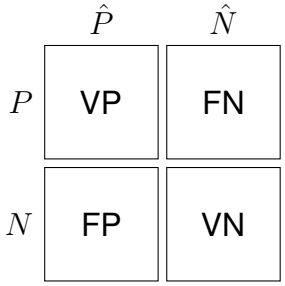
4.5.1 Matriz de confusão

A matriz de confusão permite visualizar o desempenho dos algoritmos de classificação separando os resultados em quatro classes:

- Verdadeiro Positivo (VP) : Valor predito e real são positivos;
- Verdadeiro Negativo (VN) : Valor predito e real são negativos;
- Falso Positivo (FP) : Valor predito positivo e valor real negativo;
- Falso Negativo (FN) : Valor predito negativo e valor real positivo.

Essas classes são organizadas em posições padrão de uma tabela, como pode ser visualizado na Figura 16. Para este trabalho o caso positivo é onde ocorre *Knock*.

Figura 16 – Matriz de confusão



Fonte: Produzida pelo autor.

4.5.1.1 Acurácia

A acurácia é simplesmente a taxa de predições corretas, dada como:

$$\text{Acurácia} = \frac{VP + VN}{VP + FN + FP + VN} \tag{4}$$

Nota-se que para conjuntos de dados desbalanceados, como o deste trabalho, onde há uma grande diferença de ocorrências entre as classes a acurácia pode ser enganadora, por exemplo em um conjunto onde somente 1% dos casos são positivos um algoritmo que prevê sempre negativo tem 99% de acurácia. Para abordar este problema, se utilizam Precisão e *Recall*.

4.5.1.2 Precisão

A Precisão pode ser definida como a taxa de previsões positivas que estavam realmente corretas e é calculada por:

$$\text{Precisão} = \frac{VP}{VP + FP} \tag{5}$$

4.5.1.3 Recall

O *Recall* pode ser definido como a taxa de todos os casos que realmente eram positivos que foram detectados como positivos e é calculado por:

$$\textit{Recall} = \frac{VP}{VP + FN} \tag{6}$$

4.5.1.4 Pontuação f1

Precisão e *Recall* geralmente estão em conflito, ao ajustar os parâmetros de um modelo é possível aumentar um ao custo do outro e este balanço depende da aplicação. Porém, a Pontuação f1 é uma medida que incorpora ambos e permite uma noção geral de desempenho, calculada por:

$$\text{Pontuação f1} = \frac{2}{(1/\text{Precisão}) + (1/\textit{Recall})} \tag{7}$$

5 AVALIAÇÃO

Após a implementação de todo o fluxo de processamento de dados, treino, teste e metrificação para todos os algoritmos, é preciso coletar os dados significativos aos fenômeno. Os dados são coletados dos experimentos, inseridos no fluxo e as métricas de Precisão, *Recall* e Pontuação f1 foram escolhidas para a comparação entre os algoritmos, pois são mais representativas do desempenho real e possuem uma interpretação probabilística, ao contrário de se utilizar uma função de perda arbitrária. Na Tabela 2 encontram-se as métricas para todos os algoritmos, as quais serão discutidas nas seções seguintes.

Tabela 2 – Comparação entre os algoritmos propostos

Nome	Precisão	Recall	Pontuação f1
Modelo Aleatório	0%	0%	0%
Classificador de Sequências	41%	41%	41%
AutoEncoder Denso	36%	33%	35%
AutoEncoder Convolucional	6%	45%	10%
Classificador com Extração de variáveis	74%	77%	76%

5.1 DADOS AQUISITADOS

Os conjuntos de dados mostrados a seguir foram gerados a partir do processo experimental detalhado no Capítulo 4, porém em condições levemente diferentes, além de durações diferentes, como visto na Tabela 3.

Tabela 3 – Duração dos experimentos

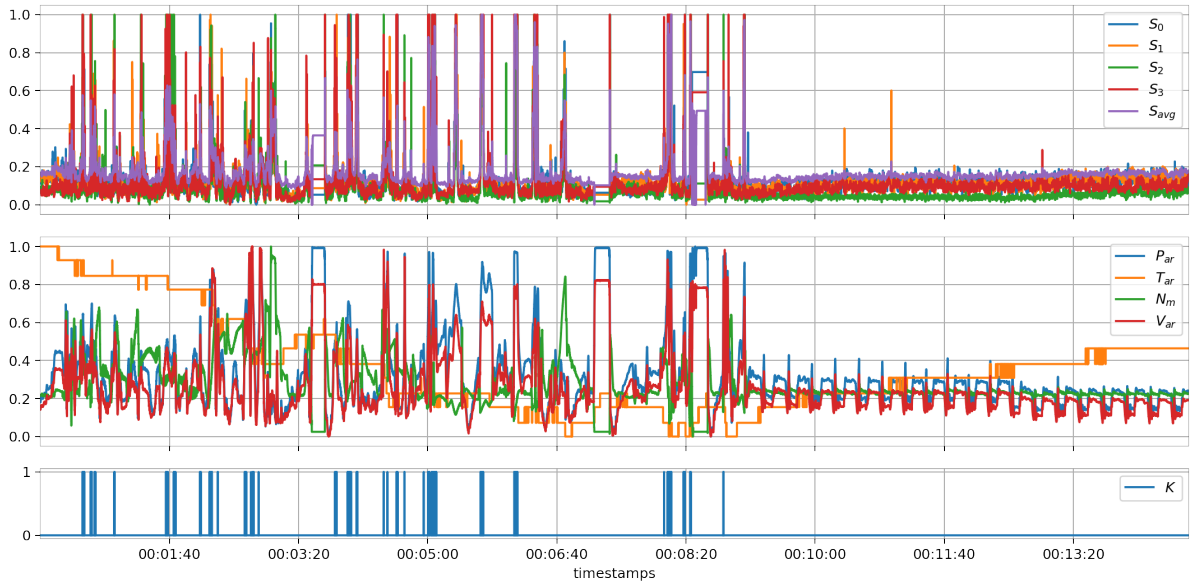
Experimento	Duração	Nº de *Knocks*
1	00:03:59	21
2	00:14:50	323
3	00:01:48	19
4	00:30:12	672
5	00:17:54	63
6	00:34:43	180

Todos os experimentos foram realizados com o carro em movimento, mas com diferentes padrões de aceleração e troca de marcha, assim, todos os experimentos foram concatenados para as fases de treino e teste, fornecendo uma variedade de modos de operação para alimentar os algoritmos.

Os dados plotados nas figuras do Apêndice A e na Figura 17 já sofreram a

etapa de pré-processamento, logo, as variáveis contínuas estão normalizadas entre $[0, 1]$ e os contadores estão diferenciados. No primeiro gráfico dos experimentos estão plotadas as variáveis de vibração (*Knock Noise*) S, no segundo as outras variáveis contínuas e por fim no terceiro a variável K, a detecção de *Knock* pela ECU em qualquer um dos cilindros. A Figura 17 é um exemplo do Apêndice A e ao analisá-la já é possível notar a alta correlação entre as variáveis S e K, e a clara separação dos períodos saudáveis e anômalos.

Figura 17 – Dados do Experimento 2

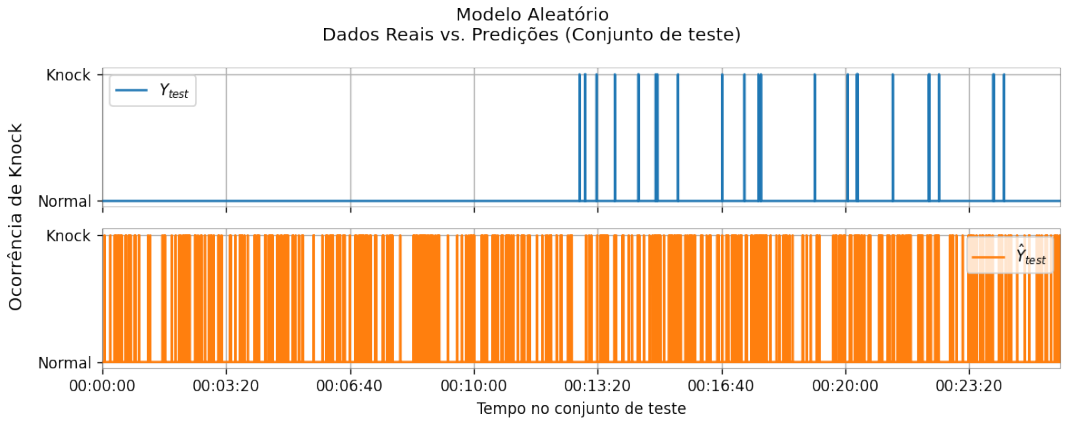


Fonte: Produzida pelo autor.

5.2 MODELO ALEATÓRIO

Dada a natureza o modelo, o número de falhas é aproximadamente o mesmo em Y_test e $\hat{Y}_test$ e por esse número ser pequeno a acurácia é aproximadamente 100%, porém como pode se observar na Tabela 2 e na Figura 18 apesar disto, este modelo não possui bom desempenho, o desempenho é melhor refletido pela pontuação f1 de 0%. Apesar disso, este modelo é útil demonstração das métricas.

Figura 18 – Predições do Modelo Aleatório



Fonte: Produzida pelo autor.

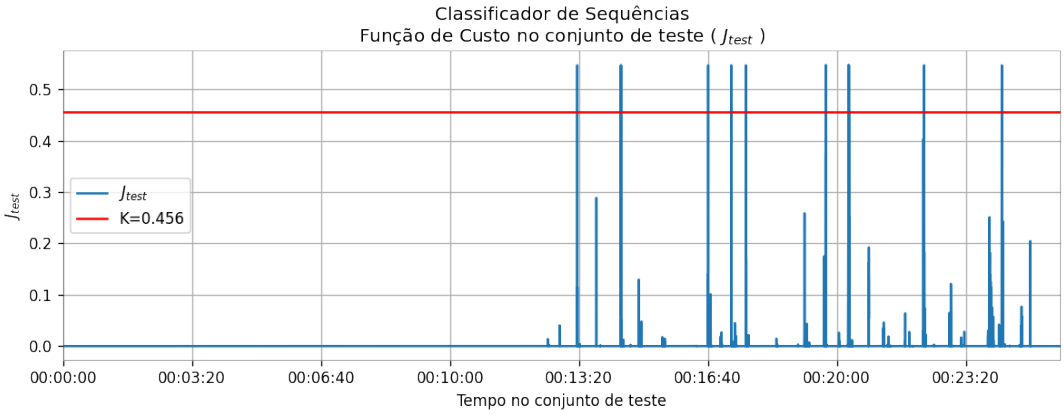
5.3 CLASSIFICADOR DE SEQUÊNCIAS

Apesar do pré-processamento para gerar as sequências ser relativamente complexo, este modelo aplica os fundamentos básicos das RNAs e seu resultado é descrito na Tabela 2. A arquitetura utilizada nos experimentos está descrita na Figura 21. Para se obter as previsões, os valores da função de custo do modelo acima de um limite K são consideradas falhas, como observado na Figura 19 e K é escolhido para que maximize a pontuação f1.

Porém, ao se observar a Figura 20, nota-se que as previsões do modelo (laranja) estão próximas das detecção da ECU (azul), apesar da baixa pontuação f1. Isso pode ser atribuído ao fato de que as métricas são calculadas ponto a ponto, ou seja, se o algoritmo detectar o *Knock* com uma defasagem no tempo, a previsão contará como falso. Dependendo da aplicação isso pode ou não ser considerado aceitável.

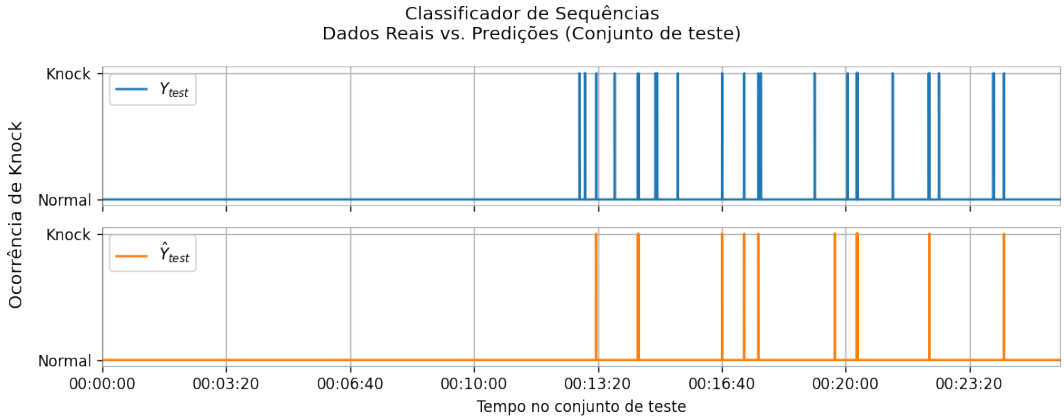
Além disso, é possível que o modelo não tenha capacidade de extrair características mais abstratas da série temporal somente com as interações de soma ponderada e função de ativação entre os neurônios artificiais, apesar da grande quantidade de parâmetros treináveis, necessitando de algo como uma convolução.

Figura 19 – Função de custo do Classificador de Sequências



Fonte: Produzida pelo autor.

Figura 20 – Predições do Classificador de Sequências



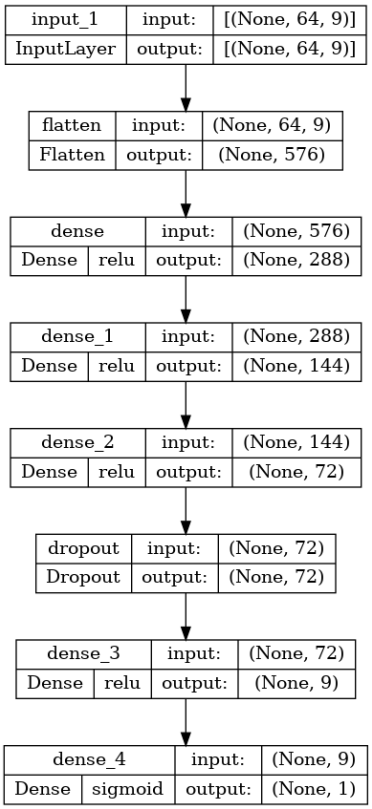
Fonte: Produzida pelo autor.

5.4 AUTOENCODER DENSO

Este modelo implementa a arquitetura básica de um AutoEncoder, descrita em detalhes na Figura 24, utilizando apenas um ponto no tempo ao invés de um sequência de pontos como os outros modelos. Apesar disso, ele consegue resultados similares ao Classificador de Sequências com pontuação f1 de 35%. Além disso, uma das vantagens dele é possuir uma quantidade baixa de parâmetros e um passo de pré-processamento mais simples, reduzindo muito requisitos de memória e processamento para o treinamento e inferência.

Observando o histograma na Figura 22, pode se observar a grande similaridade entre as distribuições J_{train} (azul) e J_{test} (verde), porém J_{test} possui áreas no lado direito onde não há sobreposição com J_{train}, consequentemente, é um pouco mais baixa no lado esquerdo. Isso é devido à presença das anomalias, que o modelo não aprendeu a

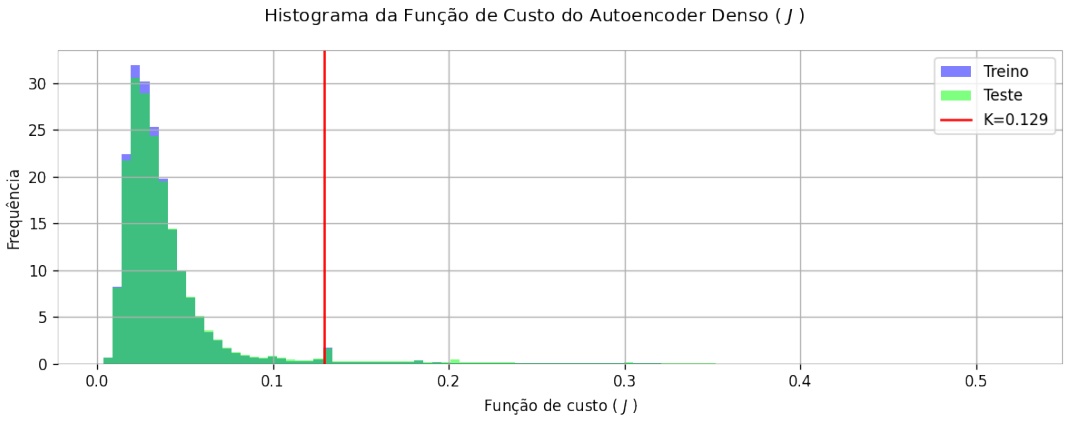
Figura 21 – Arquitetura do Classificador de Sequências



Fonte: Produzida pelo autor.

representar. Neste caso o limite para detecção de anomalia K é escolhido é o quantil 99.5% de J_{train}, aproximadamente 0.21.

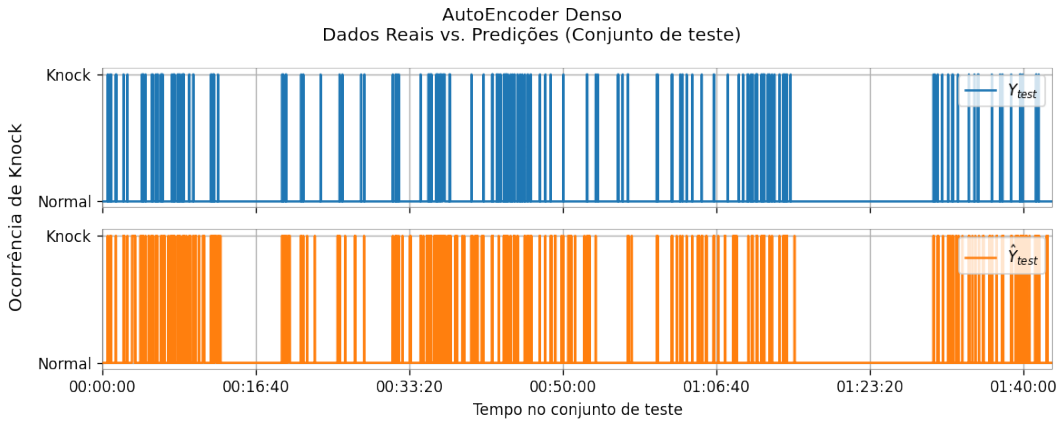
Figura 22 – Função de custo do AutoEncoder Denso



Fonte: Produzida pelo autor.

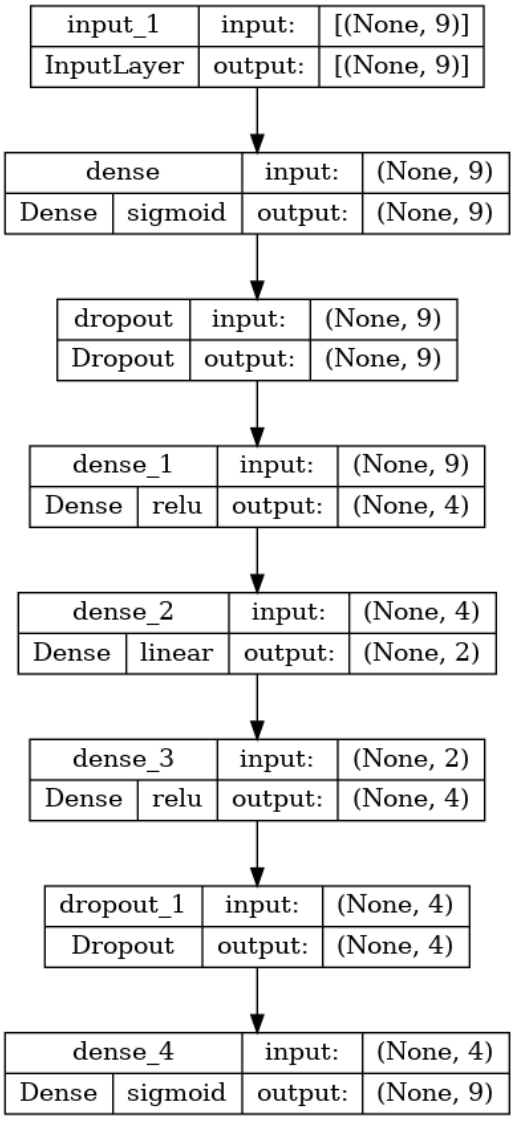
Porém a diferença não é expressiva, isso explica o mau desempenho, para uma clara divisão entre *Knock* e não-*Knock*, se espera que J_{test} possua uma protuberância maior mais à direita do histograma.

Figura 23 – Predições do AutoEncoder Denso



Fonte: Produzida pelo autor.

Figura 24 – Arquitetura do AutoEncoder Denso

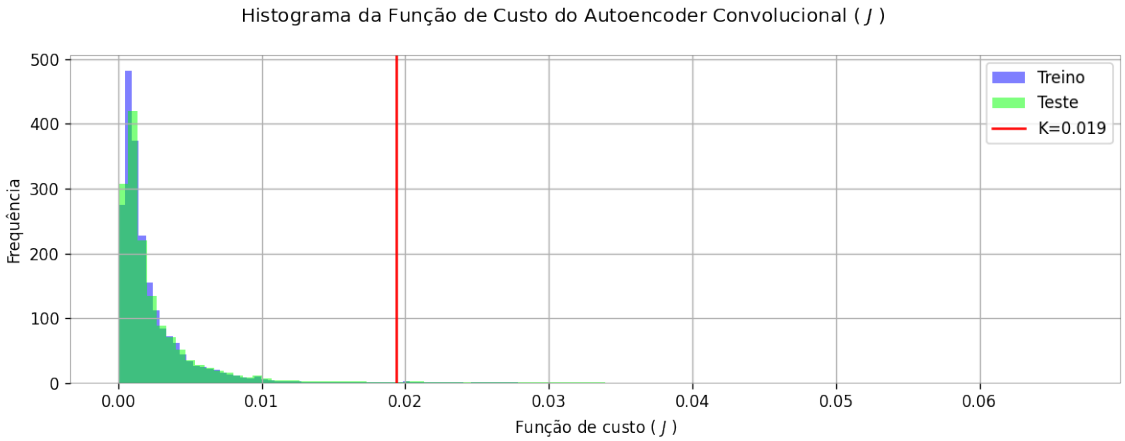


Fonte: Produzida pelo autor.

5.5 AUTOENCODER CONVOLUCIONAL

O objetivo deste modelo era solucionar os possíveis problemas do AutoEncoder Denso com uma RNC, porém ele acaba com um desempenho inferior, como observado na Tabela 2. Além disso, ele possui um passo de pré-processamento mais complexo (gerar as sequências), operações custosas para inferência (convoluções) e mais parâmetros para otimizar, como pode ser observado em detalhes na Figura 27.

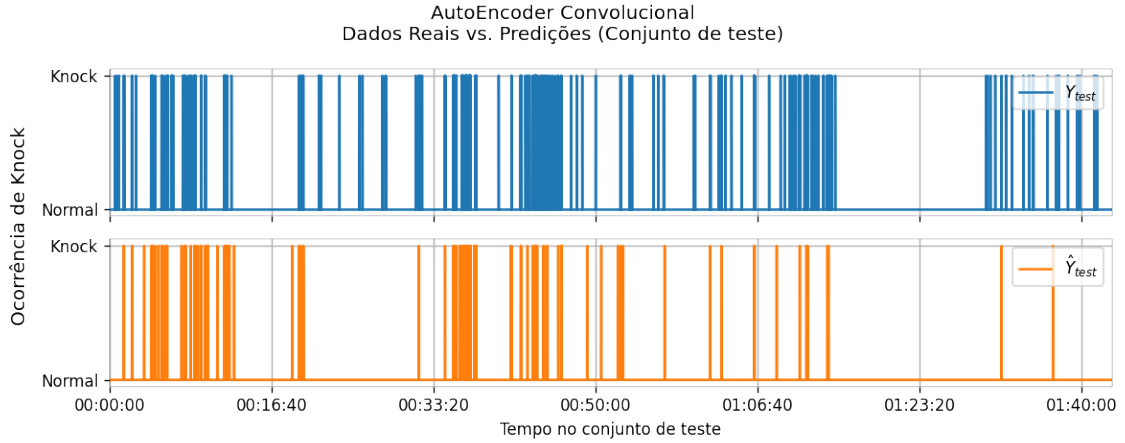
Figura 25 – Função de custo do AutoEncoder Convolucional



Fonte: Produzida pelo autor.

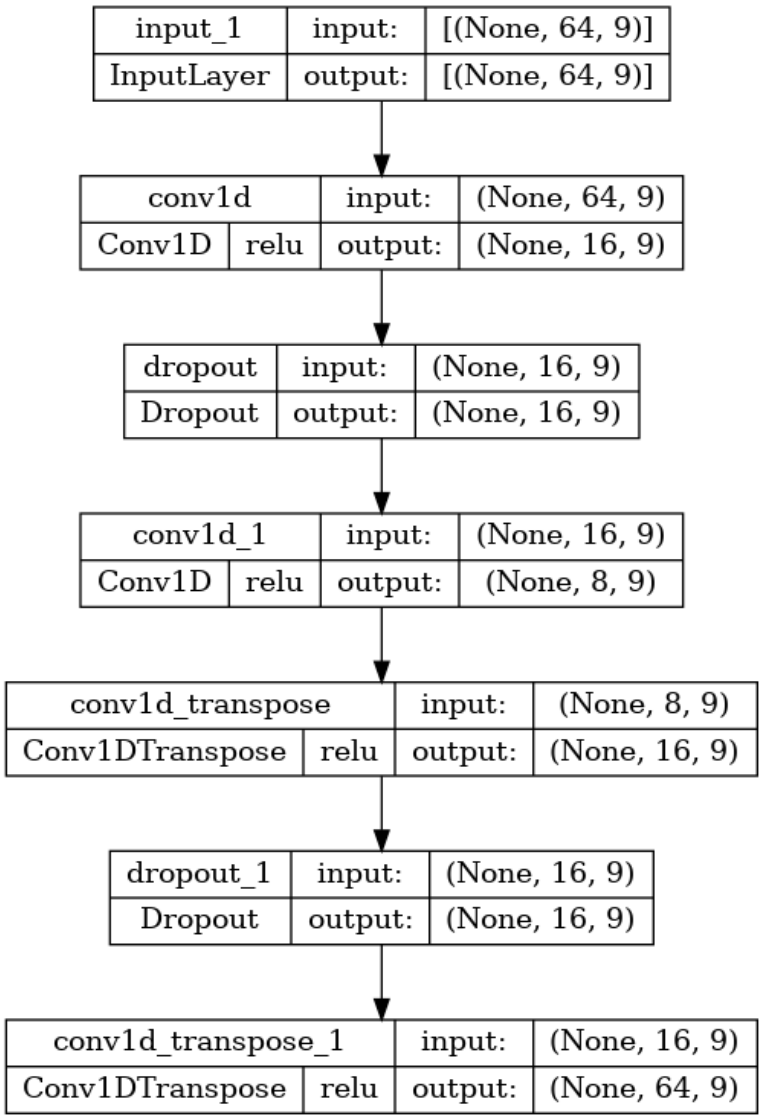
Teoriza-se que a razão do mau desempenho é que apesar do modelo não ter sido treinado com dados contendo *Knock*, ele ainda consegue reconstruir os dados anômalos, por ter mais parâmetros e usar convoluções, resultando nos histogramas de J_{train} e J_{test} muito similares e deslocados para a esquerda vistos na Figura 25.

Figura 26 – Predições do AutoEncoder Convolucional



Fonte: Produzida pelo autor.

Figura 27 – Arquitetura do AutoEncoder Convolucional

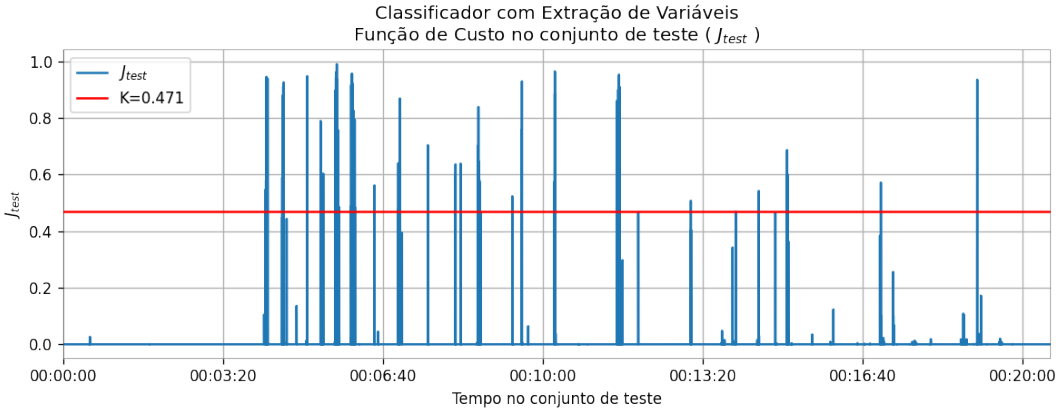


Fonte: Produzida pelo autor.

5.6 CLASSIFICADOR COM EXTRAÇÃO DE VARIÁVEIS

O objetivo deste modelo era melhorar o desempenho do Classificador de Sequências ao se adicionar uma camada convolucional, como descrito em detalhes na Figura 30, para que o modelo aprenda uma transformação dos dados que torne a fase de classificação mais efetiva em separar as sequências saudáveis e anômalas.

Figura 28 – Função de custo do Classificador com Extração de variáveis



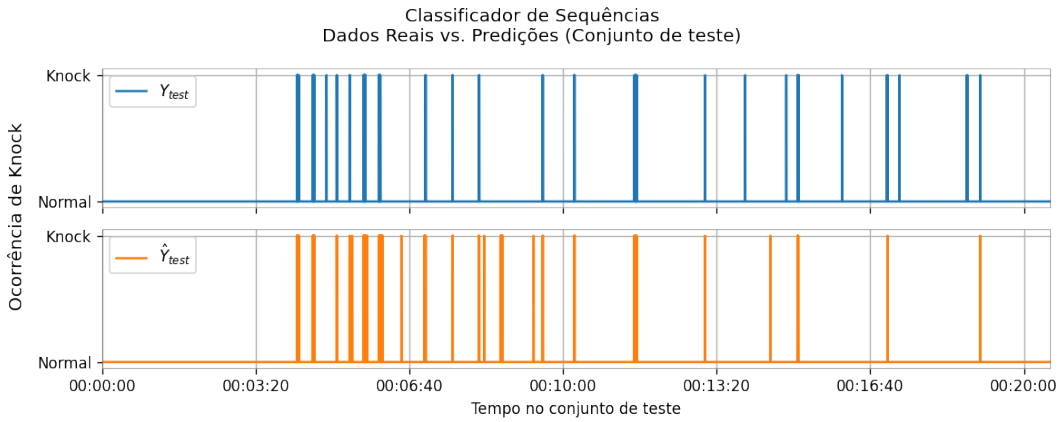
Fonte: Produzida pelo autor.

Com relativamente poucas modificações em relação ao Classificador de Sequências este modelo obtém o melhor desempenho entre os modelos propostos, com uma pontuação f1 de 76%, como visto na Tabela 2.

5.7 CONSIDERAÇÕES PARCIAIS

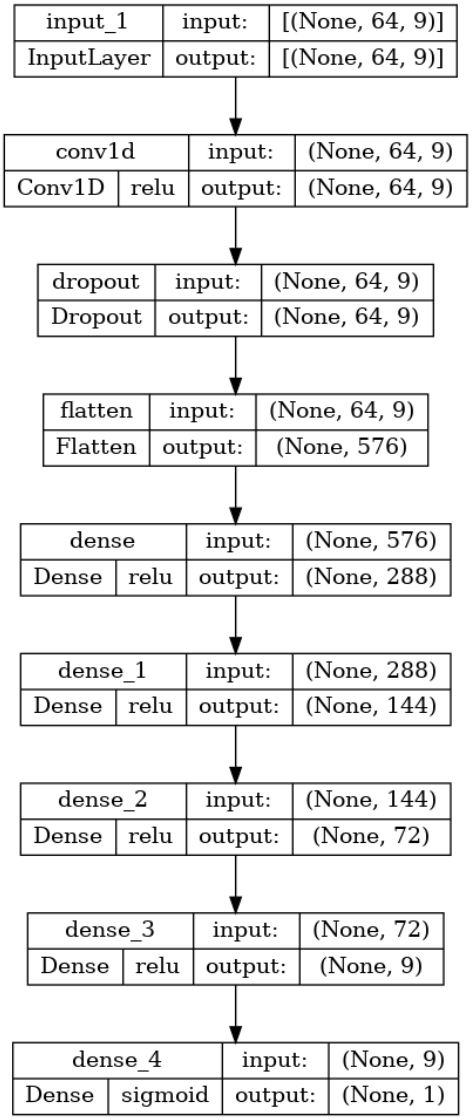
Por fim, ao se analisar as predições de cada algoritmo, o Classificador com Extração de Variáveis se prova superior para a tarefa de detecção de *Knock*. Teoriza-se

Figura 29 – Predições do Classificador com Extração de variáveis



Fonte: Produzida pelo autor.

Figura 30 – Arquitetura do Classificador com Extração de Variáveis



Fonte: Produzida pelo autor.

que isso se deve ao fato da estrutura convolucional de extração de características dos dados temporais gerar uma representação onde a tarefa de separar a classes de dados normais de anômalos se torna mais simples para o classificador, que por si só já se mostrou promissor. Além disso, o AutoEncoder Denso se mostrou promissor para conjuntos de dados Não-Supervisionados, é possível que com a otimização correta o AutoEncoder convolucional venha a se destacar nessas situações.

6 CONCLUSÕES

O crescente no uso de algoritmos de IA nas mais diversas áreas é justificada, eles podem oferecer soluções para problemas anteriormente intratáveis, otimizações para métodos existentes, ou simplesmente outra maneira de resolver desafios existentes. Mas a IA não é livre de desvantagens, as limitações computacionais, coleta de dados e projeto dos modelos são obstáculos que devem ser ultrapassados para aplicá-la. Porém, dada a infraestrutura para coletar e processar os dados é possível, com relativamente pouco conhecimento de domínio, modelar fenômenos complexos.

Com a utilização do hardware e infraestrutura da nuvem do projeto IASE, a colaboração do LISHA e da Renault, foi possível realizar os experimentos e coletar os dados necessários para a aplicação das técnicas de AM para a tarefa de detectar *Knock* em motores automotivos.

Vale lembrar que métodos de detecção de *Knock* já existem, porém podem demandar sensores caros, conhecimento de domínio para modelagem matemática, calibração fina ou altas taxas de aquisição dos dados. O objetivo deste trabalho foi estudar métodos alternativos possibilitando diminuição nos custos.

A partir somente dos dados coletados e do estudo dos fundamentos destas técnicas foram construídos RNAs que realizam a tarefa de detectar um fenômeno complexo como o *Knock* com desempenho similar à algoritmos ajustados por especialistas da Renault, os quais não possuem a limitação de amostragem imposta pela interface com a ECU.

No entanto é importante ressaltar que a qualidade de um modelo de AM só será tão boa quanto os dados de treino, então é possível que todos os modos de operação e falha não tenham sido capturados nos experimentos realizados. Além disso, o algoritmo interno da ECU foi considerado como 100% correto, é possível que haja diferenças caso fossem utilizados métodos mais precisos, como os que utilizam a pressão interna do cilindro. Além disso, há diversos parâmetros dos modelos que dependendo da aplicação podem ser ajustados, como tamanho do janelamento e sensibilidade, causando diferenças nos resultados.

Ademais, há muito espaço para otimização quanto a arquitetura de cada modelo, quantidade de neurônios artificiais e camadas podem ser ajustadas para obter diferentes perfis de desempenho e custo computacional. Entretanto, constata-se que nos algoritmos testados, a arquitetura geral do classificador foi superior e o uso de camadas convolucionais pode melhorar o desempenho na detecção. Por fim, a partir dos resultados, se conclui que o uso do aprendizado de máquina é um caminho válido a se explorar para a aplicação abordada neste trabalho.

REFERÊNCIAS

AL-ZEYADI, M. et al. Deep learning towards intelligent vehicle fault diagnosis. **IEEE**, 2020.

ALBAWI, S.; MOHAMMED, T. A.; AL-ZAWI, S. Understanding of a convolutional neural network. **IEEE**, p. 1–6, 2017.

CORNELL AERONAUTICAL LABORATORY. **The Perceptron: a perceiving and recognizing automaton**. 1957. Report 85-460-1 prepared by Frank Rosenblatt. Disponível em: https://blogs.umass.edu/brain-wars/files/2016/03/rosenblatt-1957.pdf. Acesso em: 08 fev. 2022.

FRÖHLICH, A. A. Smartdata: an iot-ready api for sensor networks. **International Journal of Sensor Networks**, Inderscience Publishers (IEL), v. 28, n. 3, p. 202–210, 2018.

GÉRON, A. **Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems**. Sebastopol, Califórnia, United States: O'Reilly Media, Inc., 2019.

HEYWOOD, J. B. **Internal combustion engine fundamentals**. [S.l.]: McGraw-Hill Education, 2018.

JORDAN, M. I.; MITCHELL, T. M. Machine learning: Trends, perspectives, and prospects. **Science**, American Association for the Advancement of Science, v. 349, n. 6245, p. 255–260, 2015.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. **The bulletin of mathematical biophysics**, Springer, v. 5, n. 4, p. 115–133, 1943.

MILLO, F.; FERRARO, C. Knock in si engines: a comparison between different techniques for detection and control. **SAE transactions**, JSTOR, p. 1091–1112, 1998.

MITCHELL, T. **Machine Learning**. New York: McGraw-Hill, 1997.

RUDER, S. An overview of gradient descent optimization algorithms. **arXiv preprint arXiv:1609.04747**, 2016.

SHAHID, S. M.; KO, S.; KWON, S. Real-time abnormality detection and classification in diesel engine operations with convolutional neural network. **Expert Systems with Applications**, v. 192, p. 116233, 2022. ISSN 0957-4174. Disponível em: https://www.sciencedirect.com/science/article/pii/S0957417421015451.

SHARMA, D.; KUMAR, N. A review on machine learning algorithms, tasks and applications. **International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)**, v. 6, n. 10, p. 2278–1323, 2017.

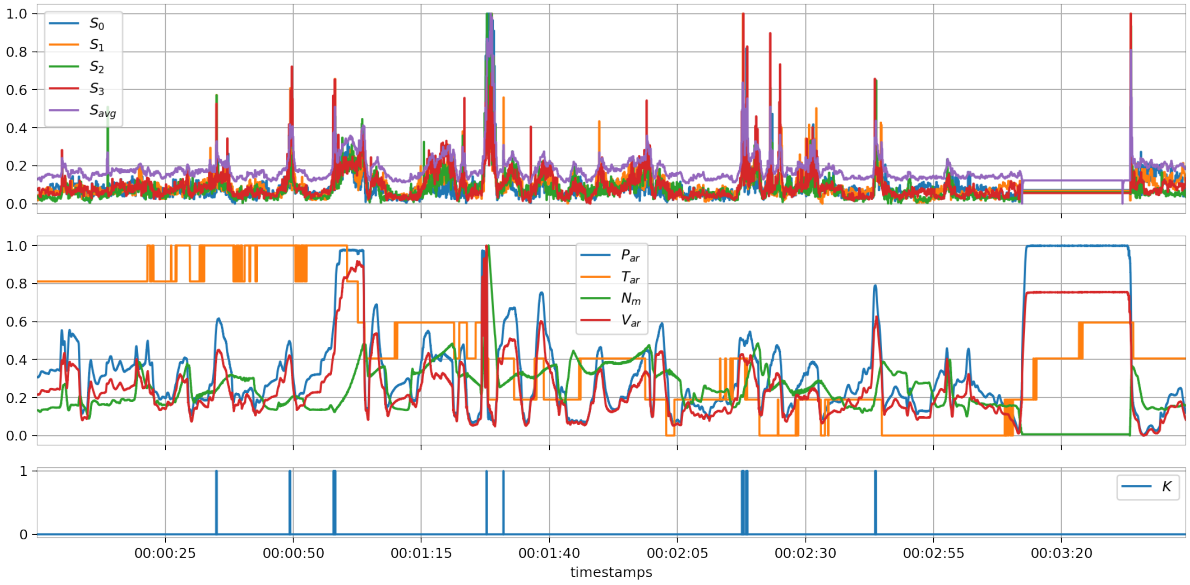
SHARMA, N. **Knock model evaluation - gas engine**. Master of Science Thesis (Machine Design), KTH Industrial Engineering and Management, Stockholm, Sweden, 2018.

YAN, W. Detecting gas turbine combustor anomalies using semi-supervised anomaly detection with deep representation learning. **Cognitive Computation**, Springer, v. 12, n. 2, p. 398–411, 2020.

ZHEN, X. et al. The engine knock analysis: An overview. **Applied Energy**, v. 92, p. 628–636, 2012.

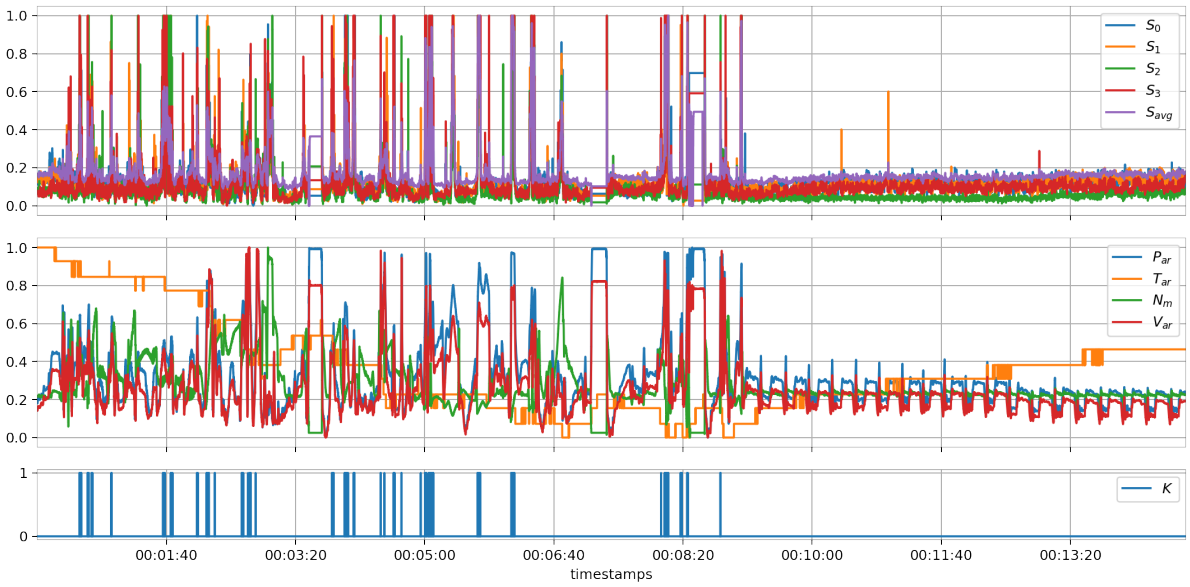
APÊNDICE A - DADOS COLETADOS

Figura 31 – Dados do Experimento 1



Fonte: Produzida pelo autor.

Figura 32 – Dados do Experimento 2

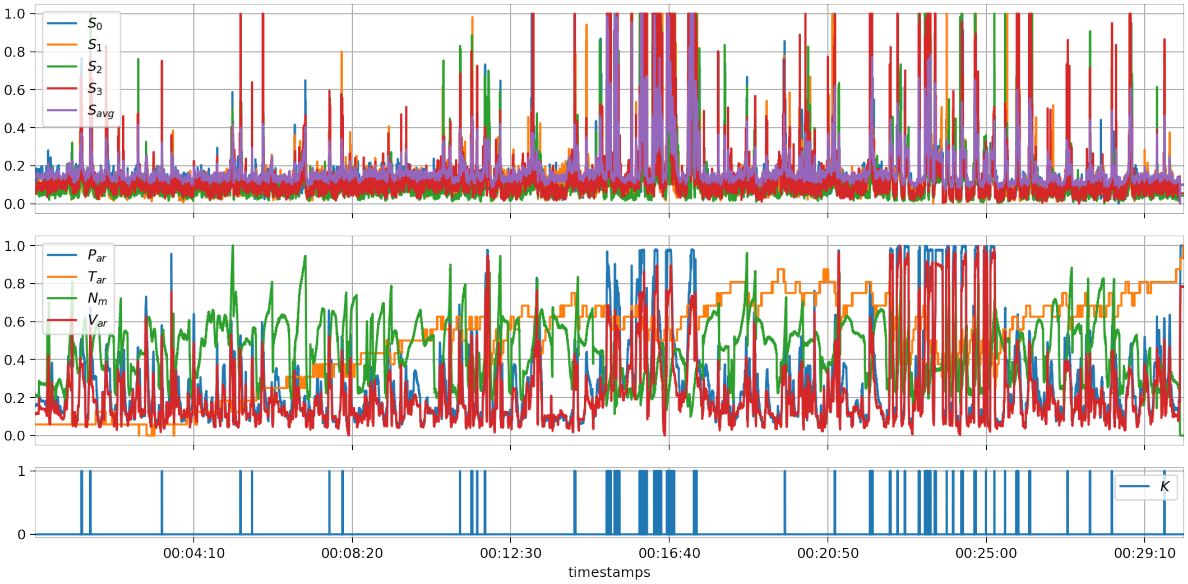


Fonte: Produzida pelo autor.

Figura 33 – Dados do Experimento 3

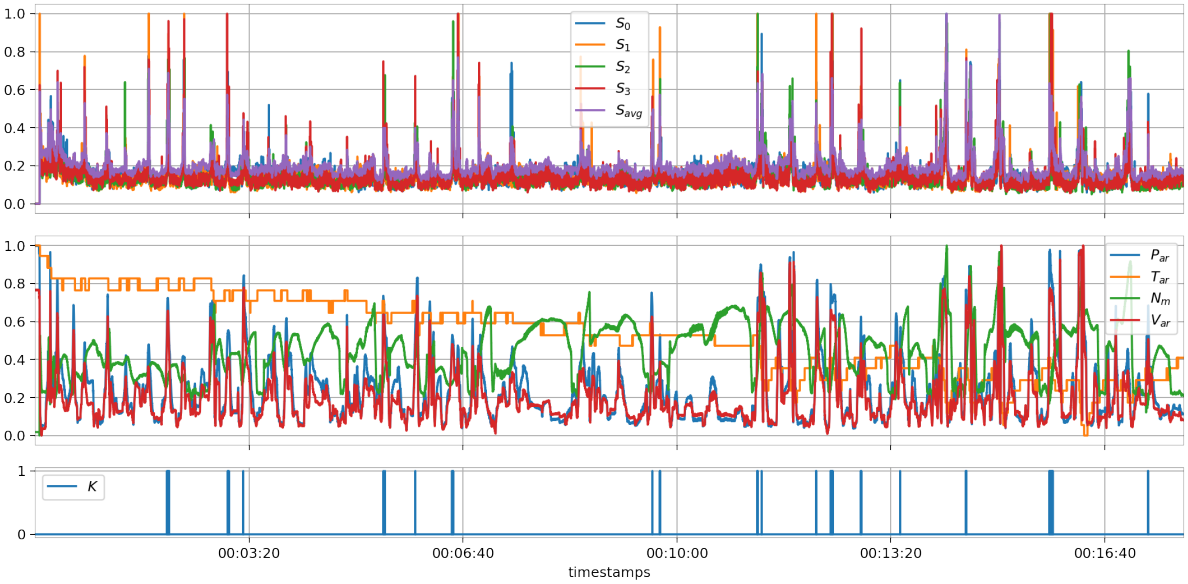
Fonte: Produzida pelo autor.

Figura 34 – Dados do Experimento 4



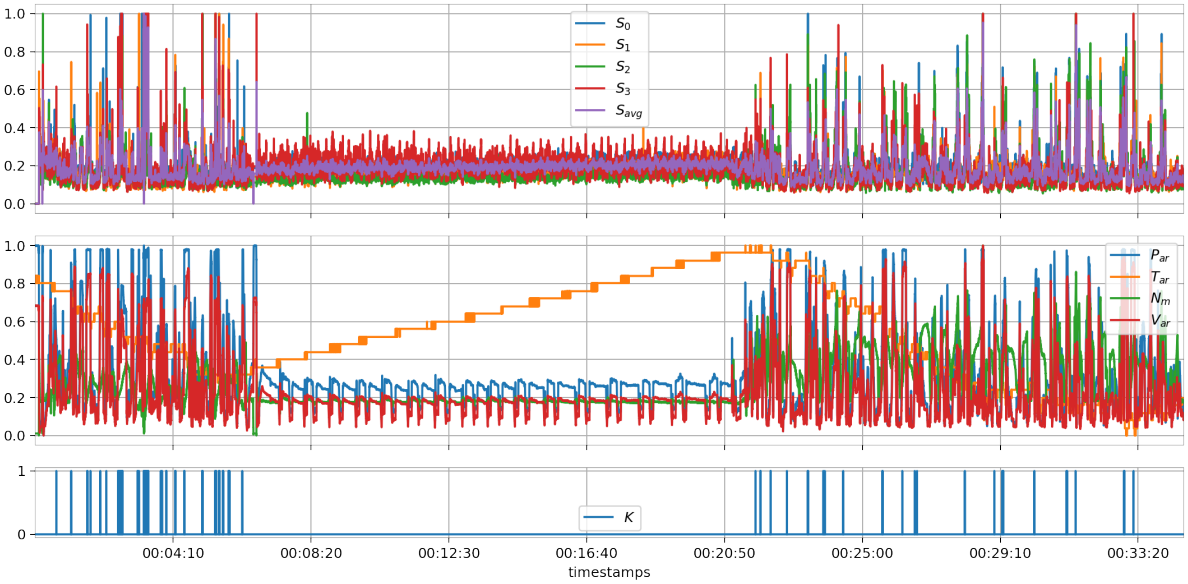
Fonte: Produzida pelo autor.

Figura 35 – Dados do Experimento 5



Fonte: Produzida pelo autor.

Figura 36 – Dados do Experimento 6



Fonte: Produzida pelo autor.