Universidade Estadual do Paraná
Campus Apucarana

GIAN ROBERTO PEREIRA DA SILVA

SÍNTESE DE SÉRIES TEMPORAIS UTILIZANDO AUTOENCODERS PARA GERAÇÃO DE DADOS SINTÉTICOS REALISTAS

APUCARANA–PR

2025

GIAN ROBERTO PEREIRA DA SILVA

SÍNTESE DE SÉRIES TEMPORAIS UTILIZANDO AUTOENCODERS PARA GERAÇÃO DE DADOS SINTÉTICOS REALISTAS

Trabalho de Conclusão de Curso apresentado ao curso de Bacharelado em Ciência da Computação da Universidade Estadual do Paraná para obtenção do título de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. José Luís Seixas Junior

APUCARANA–PR

2025

Ficha catalográfica elaborada pelo Sistema de Bibliotecas da UNESPAR e
Núcleo de Tecnologia de Informação da UNESPAR, com Créditos para o ICMC/USP
e dados fornecidos pelo(a) autor(a).

Roberto Pereira da Silva, Gian
   Síntese de Séries Temporais utilizando
AutoEncoders para Geração de Dados Sintéticos
Realistas / Gian Roberto Pereira da Silva. --
Apucarana-PR,2025.
   51 f.: il.

   Orientador: José Luís Seixas Junior.
   Trabalho de Conclusão de Curso, Ciência da
Computação - Universidade Estadual do Paraná, 2025.

   1. AutoEncoders. 2. Aprendizado de Máquina. I -
Luís Seixas Junior, José (orient). II - Título.

Gian Roberto Pereira da Silva

Síntese de Séries Temporais utilizando AutoEncoders para Geração de Dados Sintéticos Realistas/ Gian Roberto Pereira da Silva. – Apucarana–PR, 2025-

51 p. : il. algumas color. ; 30 cm.

Orientador: Prof. Dr. José Luís Seixas Junior

– Universidade Estadual do Paraná, 2025.

1. Séries Temporais. 2. AutoEncoders. I. José Luís Seixas Junior. II. Universidade Estadual do Paraná. III. Faculdade de Apucarana. IV. Síntese de Séries Temporais utilizando AutoEncoders para Geração de Dados Sintéticos Realistas

CDU 02:141:005.7

GIAN ROBERTO PEREIRA DA SILVA

SÍNTESE DE SÉRIES TEMPORAIS UTILIZANDO AUTOENCODERS PARA GERAÇÃO DE DADOS SINTÉTICOS REALISTAS

Trabalho de Conclusão de Curso apresentado ao curso de Bacharelado em Ciência da Computação da Universidade Estadual do Paraná para obtenção do título de Bacharel em Ciência da Computação.

BANCA EXAMINADORA

Prof. Dr. José Luís Seixas Junior
Universidade Estadual do Paraná
Orientador

Prof. Dr. Paulo Roberto de Oliveira
Universidade Estadual do Paraná

Prof. Dr. Kleber Márcio de Souza
Universidade Estadual do Paraná

Apucarana–PR, 17 de dezembro de 2025

Este trabalho é dedicado as pessoas que usam a Computação como ferramenta de trabalho.

AGRADECIMENTOS

Agradeço primeiramente ao meu orientador, Prof. Dr. José Luis Seixas Junior, pela paciência, dedicação e pelas discussões que foram essenciais para a definição e o refinamento deste trabalho. Sua orientação foi fundamental em todas as etapas do projeto.

E agradeço também ao meu colega Fabrício Pereira Diniz que me auxiliou na correção deste trabalho. Sua ajuda foi de muita importância para que eu conseguisse minimizar os meus erros.

"O que é real? Como você define o 'real'?"

— Morpheus (Matrix)

SILVA, GIAN ROBERTO PEREIRA DA. **Síntese de Séries Temporais utilizando AutoEncoders para Geração de Dados Sintéticos Realistas**. 51 p. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Universidade Estadual do Paraná, Apucarana–PR, 2025.

RESUMO

A geração de dados sintéticos de séries temporais é uma solução crucial para mitigar a escassez de dados em domínios como indústria, finanças e saúde. Este trabalho propõe e valida uma metodologia baseada em *Autoencoders* (AE) para a síntese de dados realistas e úteis. Focando em um conjunto de dados complexo de telemetria de propulsores de espaçonaves, que exibe características ruidosas e em surto, a pesquisa demonstra uma progressão metodológica. Inicialmente, um AutoEncoder Densa unicanal é avaliado como *baseline*, demonstrando-se incapaz de capturar a dinâmica temporal DTW de 14.22 e a distribuição estatística Wasserstein de 0.2647 dos eventos, gerando apenas ruído. Esta falha motiva a arquitetura proposta: um *Autoencoder* Convolucional 1D CNN multicanal, que modela a relação física de causa-e-efeito entre o comando de acionamento `ton` e a resposta do propulsor `thrust`, `mfr`. A avaliação é realizada com um protocolo robusto DTW, Distância de Wasserstein, t-SNE e uma métrica de utilidade TSTR - *Train-on-Synthetic, Test-on-Real*. Os resultados demonstram a superioridade da CNN, que reduziu o erro de forma DTW para 7.30 e replicou a distribuição de dados Wasserstein de 0.044. Notavelmente, o TSTR atingiu um *ratio* de 0.0192, validando o AE como um filtro de ruído eficaz e um gerador de dados de alta fidelidade.

Palavras-chave: Séries Temporais. AutoEncoders. Dados Sintéticos. Aprendizado de Máquina. Aprendizado Profundo.

SILVA, GIAN ROBERTO PEREIRA DA. **Time Series Synthesis using AutoEncoders for Realistic Synthetic Data Generation**. 51 p. Final Project (Bachelor of Science in Computer Science) – State University of Paraná, Apucarana–PR, 2025.

ABSTRACT

The generation of synthetic time series data is a crucial solution to mitigate data scarcity in domains such as industry, finance, and healthcare. This work proposes and validates a methodology based on Autoencoders for the synthesis of realistic and useful data. Focusing on a complex spacecraft thruster telemetry dataset, characterized by noisy and burst-like behaviors, the research demonstrates a methodological progression. Initially, a single-channel Dense MLP AE is evaluated as a *baseline*, showing an inability to capture temporal dynamics DTW of 14.22 and the statistical distribution Wasserstein of 0.2647 of the events, producing only noise. This shortcoming motivates the proposed architecture: a multi-channel 1D Convolutional Autoencoder CNN, which models the physical cause-and-effect relationship between the actuation command `ton` and the thruster response `thrust`, `mfr`. The evaluation follows a robust protocol DTW, Wasserstein Distance, t-SNE and includes a utility metric TSTR – *Train-on-Synthetic, Test-on-Real*. The results demonstrate the superiority of the CNN, which reduced the shape error DTW to 7.30 and replicated the data distribution Wasserstein of 0.044. Notably, the TSTR achieved a ratio of 0.0192, showing that the generated synthetic data are 52 times more useful for a forecasting task than the noisy real data, thus validating the AE as both an effective noise filter and a high-fidelity data generator.

Keywords: Time Series. AutoEncoders. Synthetic Data. Machine Learning. Deep Learning.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AE	AutoEncoder
ARIMA	Autoregressive Integrated Moving Average
BCE	Binary Cross-Entropy
CNN	Convolutional Neural Network
DTW	Dynamic Time Warping
GAN	Generative Adversarial Network
GRU	Gated Recurrent Unit
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
ML	Machine Learning
MLP	Multilayer Perceptron
MSE	Erro Quadrático Médio
RNN	Rede Neural Recorrente
TSTR	Train-on-Synthetic, Test-on-Real

SUMÁRIO

1 INTRODUÇÃO

O advento do *Big Data* e o avanço exponencial do poder computacional impulsionaram o Aprendizado de Máquina *Machine Learning* (ML) para o centro da inovação tecnológica. Modelos de *Deep Learning* são agora capazes de resolver problemas de alta complexidade, desde o reconhecimento de imagens até a tradução de idiomas. No entanto, o desempenho desses modelos é fundamentalmente dependente da qualidade e, principalmente, da quantidade de dados disponíveis para treinamento. Em muitos domínios críticos—como o diagnóstico médico, a detecção de fraudes financeiras ou o monitoramento de sistemas industriais complexos—os dados são escassos, caros, ou protegidos por rigorosas leis de privacidade [1].

Essa dependência de grandes volumes de dados revela uma limitação crítica para a aplicação prática de ML. A escassez de dados representa um dos principais gargalos para o avanço da área, particularmente quando se trata de séries temporais, que capturam a evolução de um sistema ao longo do tempo. A coleta desse tipo de dado é frequentemente problemática por razões tanto técnicas quanto regulatórias: dados de sensores industriais podem ser sigilosos, dados de pacientes são confidenciais Lei Geral de Proteção de Dados e eventos de interesse, como falhas de equipamento ou anomalias financeiras, são, por definição, raros [2]. Consequentemente, essa carência de dados, especialmente de eventos raros, leva a modelos de ML com dificuldade de generalização e incapazes de prever falhas críticas.

Diante desse cenário, a Geração de Dados Sintéticos (GDS) emerge como uma solução estratégica e poderosa. Em vez de depender exclusivamente da coleta de dados reais, a GDS utilizando modelos generativos os utiliza para aprender a distribuição estatística e as dependências temporais de um conjunto de dados existente. Uma vez treinado, o modelo pode gerar novas amostras de dados que são estatisticamente idênticas aos dados reais, mas totalmente sintéticas e anônimas. Essa abordagem permite simultaneamente a criação de conjuntos de dados vastos e diversificados, a proteção da privacidade de indivíduos e a simulação de cenários de eventos raros para testes de robustez de sistemas [3].

Dentre as diversas técnicas disponíveis para GDS, os *Autoencoders* (AE) destacam-se como uma das ferramentas mais comuns para a geração de séries temporais sintéticas. Um AE é um tipo de rede neural não supervisionada [1] que aprende a realizar duas tarefas complementares: primeiro, comprimir os dados de entrada em uma representação de baixa dimensão, chamada de espaço latente, capturando apenas a essência da informação [4, 5]; segundo, descomprimir essa representação de volta ao formato original. Esta arquitetura de codificador-decodificador também é a base para modelos de sequência modernos

[6]. O espaço latente, portanto, torna-se um mapa das características fundamentais dos dados. Ao amostrar pontos desse espaço e passá-los pelo decodificador, é possível gerar dados novos e realistas que compartilham as mesmas características essenciais dos dados originais.

Este trabalho explora precisamente a aplicação de *Autoencoders* para a geração de dados sintéticos realistas de séries temporais, uma arquitetura já explorada com sucesso para tarefas em séries temporais, como a detecção de anomalias [7]. Embora abordagens como Redes Adversariais Generativas (GANs) [8] sejam populares, inclusive com adaptações específicas para séries temporais [2, 3, 9], os *Autoencoders* oferecem vantagens [1] significativas para este domínio, como um treinamento notavelmente mais estável—evitando problemas como o colapso de modo, onde o gerador aprende a produzir apenas um tipo de amostra—e um espaço latente mais interpretável. Contudo, o desafio central não é apenas gerar dados, mas provar rigorosamente que os dados gerados são realistas, utilizando métricas de similaridade apropriadas [10, 11], e, mais importante, úteis para aplicações práticas de ML, cuja qualidade pode ser inspecionada visualmente por técnicas como t-SNE [12].

O objetivo geral deste trabalho é desenvolver e validar um *pipeline* metodológico robusto para a geração de dados sintéticos de séries temporais. O foco da pesquisa é investigar a eficácia de diferentes arquiteturas de *Autoencoder* Densa vs. 1D-CNN e estabelecer um protocolo de avaliação rigoroso baseado em DTW, Wasserstein e TSTR capaz de quantificar o realismo e a utilidade dos dados gerados, superando as limitações das métricas de erro tradicionais.

Para alcançar o objetivo geral, os seguintes objetivos específicos são definidos:

Analisar as limitações fundamentais de *Autoencoders* densos (baseline) na captura de dependências temporais complexas em séries unicanais.

Propor uma arquitetura de *Autoencoder* Convolucional 1D (1D-CNN) adaptada para a geração de séries temporais multivaloradas, investigando sua capacidade de modelar simultaneamente correlações espaciais entre canais e temporais.

Definir um protocolo de avaliação multidimensional, baseado em métricas de similaridade de distribuição utilizando a Distância de Wasserstein, similaridade morfológica utilizando o *DFW* e utilidade em tarefas de ML com a métrica de *TSTR*, para validar rigorosamente a fidelidade e o realismo de séries temporais sintéticas.

Comparar o desempenho da arquitetura convolucional (CNN) com o modelo denso (baseline), quantificando a superioridade do modelo CNN na geração de dados sintéticos realistas, conforme mensurado pelo protocolo de avaliação estabelecido.

A estrutura deste trabalho está organizada para apresentar de forma sequencial o problema, as soluções propostas e a validação.

O Capítulo 2, Fundamentação Teórica, abrange os conceitos essenciais que sustentam a metodologia, detalhando as Redes Neurais [13, 14, 15], as Redes Neurais Recorrentes (RNNs) e suas variantes avançadas (LSTM e GRU) [16, 6], e a teoria de Modelos Generativos, focando especificamente nos *Autoencoders* [4, 1].

O Capítulo 3, Método de Pesquisa, detalha a metodologia adotada, incluindo a caracterização do conjunto de dados de telemetria de propulsores, a evolução do pré-processamento (Unicanal para Multicanal), a modelagem comparativa das arquiteturas *Autoencoder* Densa e 1D-CNN Multicanal, e a justificativa do protocolo de avaliação com as seguintes métricas: Distância de Wasserstein, DTW, TSTR, t-SNE [11, 10, 12].

Por fim, o Capítulo 4, Experimentos e Resultados, apresenta a análise comparativa dos modelos, diagnosticando as principais diferenças da implementação do *baseline* Densa [8] e validando a superioridade da arquitetura 1D-CNN Multicanal em todas as métricas de fidelidade e utilidade, e o Capítulo 5, Conclusão, sumariza as contribuições do trabalho e sugere direções para pesquisas futuras, como a implementação de *Autoencoders* Condicionais e a comparação com GANs [2].

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta os conceitos teóricos que fundamentam o desenvolvimento deste trabalho, abrangendo as principais arquiteturas de redes neurais utilizadas para o processamento de sequências temporais, bem como os fundamentos dos modelos generativos aplicados à geração de dados sintéticos. São abordados aspectos conceituais, avanços históricos e aplicações práticas das arquiteturas mais relevantes no contexto de aprendizado profundo.

2.1 Redes Neurais

As Redes Neurais Artificiais são modelos computacionais bio-inspirados, conceitualmente baseados na estrutura e funcionamento do cérebro humano, projetados para identificar padrões complexos e aprender a partir de dados [13, 14]. A unidade de processamento fundamental desses modelos é o neurônio artificial ou um *perceptron*, que recebe múltiplos sinais de entrada, aplica pesos sinápticos a eles, soma os resultados ponderados juntamente com um viés, ou *bias* e, em seguida, aplica uma função de ativação não linear para produzir um sinal de saída [17].

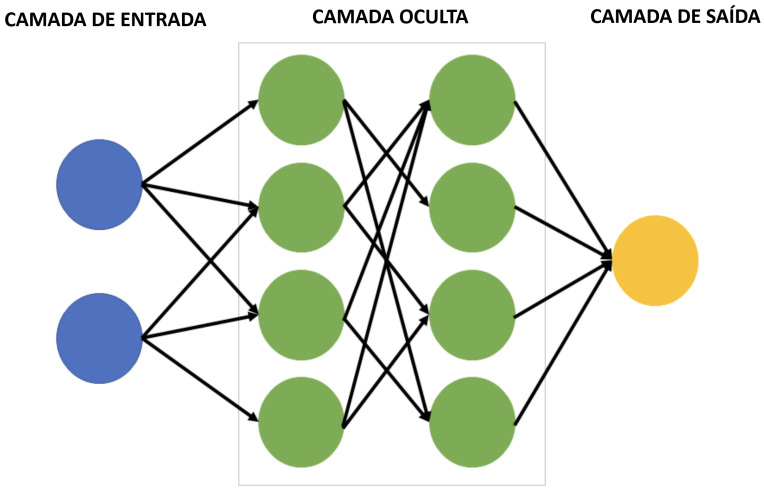

Figura 1 – Diagrama de uma RNA.

Fonte: Adaptado de [18], Autor: TseKiChun.

Uma arquitetura muito comum como mostrado na Figura 1[1] é a *feedforward*, também conhecida como *Multi-Layer Perceptron* [15]. Nesses modelos, os neurônios são organizados em camadas sucessivas. Tipicamente, essa arquitetura é composta por uma

[1] TSEKICHUN. Neural Network Diagram. Science Learn. Disponível em <https://www.sciencelearn.org.nz/images/5156-neural-network-diagram>. Acesso em: 15 nov 2025.

Camada de Entrada (*Input Layer*), que recebe os dados brutos; uma ou mais Camadas Ocultas (*Hidden Layers*), responsáveis pelo processamento e extração de características em níveis crescentes de abstração e uma Camada de Saída (*Output Layer*), que produz o resultado final da rede, como uma classificação ou regressão. O termo *feedforward* (avanço direto) indica que o fluxo de informação é unidirecional, onde os dados se propagam da camada de entrada, através das camadas ocultas, até a camada de saída, sem a existência de ciclos ou conexões retroativas. Esta é a característica que distingue as redes *feedforward* das Redes Neurais Recorrentes, que serão abordadas.

O processo de treinamento de uma rede neural *feedforward* é tipicamente realizado através do algoritmo de retropropagação (*backpropagation*) [15]. Esse processo inicia-se com uma propagação *forward*, onde a entrada é passada pela rede para gerar uma previsão. Em seguida, a discrepância entre a previsão da rede e o valor real esperado—quantificada por uma função de perda (*loss function*) é calculada. Na fase de propagação *backward*, o algoritmo de *backpropagation* utiliza a regra da cadeia do cálculo diferencial para determinar o gradiente da função de perda em relação a cada peso na rede. Por fim, ocorre a atualização dos pesos, que são ajustados na direção oposta ao gradiente, geralmente através de um método de otimização como o *Stochastic Gradient Descent* (SGD) ou suas variantes (e.g., Adam, RMSProp), com o objetivo de minimizar o erro total.

Embora as redes *feedforward* sejam grandes aproximadores universais de funções, como demonstrado por Hornik et al. [19], elas tratam cada entrada de forma independente. Elas não possuem mecanismos intrínsecos de memória para reter informações sobre entradas passadas. Essa limitação as torna inferiores às Redes Neurais Recorrentes para tarefas que envolvem dados sequenciais ou temporais, onde o contexto e a ordem são cruciais. Essa lacuna motivou o desenvolvimento das arquiteturas recorrentes.

2.2 Redes Neurais Recorrentes

As Redes Neurais Recorrentes (*RNNs*) são uma das classes mais importantes de modelos para processamento de dados sequenciais [1], capazes de modelar dependências temporais e contextuais ao longo de uma série de entradas. Diferentemente das redes neurais tradicionais (*feedforward*), as RNNs mantêm um estado interno que armazena informações sobre entradas passadas, permitindo que o modelo capture relações temporais entre observações [5, 4].

Apesar dessa capacidade inovadora de manter memória temporal, o treinamento de RNNs tradicionais apresenta desafios significativos devido a problemas como o desaparecimento e a explosão do gradiente, que dificultam o aprendizado de dependências de longo prazo [20, 21]. Para mitigar essas limitações fundamentais, foram propostas arquiteturas recorrentes aprimoradas, como a *Long Short-Term Memory* (LSTM) [16] e a *Gated*

Recurrent Unit (GRU) [6]. Essas variantes introduzem mecanismos de controle denominados *gates*, que permitem selecionar quais informações devem ser armazenadas, esquecidas ou propagadas ao longo do tempo, estabilizando o fluxo do gradiente e melhorando substancialmente o aprendizado.

Mais especificamente, a arquitetura LSTM utiliza três portas principais—entrada, esquecimento e saída para controlar o fluxo de informações e o estado de memória de longo prazo. Isso permite que a rede retenha dependências relevantes por períodos extensos sem sofrer degradação de gradiente [22]. A GRU, por sua vez, combina as funções das portas em uma estrutura mais simples e computacionalmente eficiente, mantendo desempenho competitivo em tarefas diversas [23]. Consequentemente, ambas as arquiteturas têm se mostrado eficazes em modelagem de séries temporais complexas, como dados meteorológicos, sinais fisiológicos e consumo energético [24, 25, 26].

A superioridade dessas arquiteturas torna-se evidente quando comparadas a métodos tradicionais. Modelos lineares clássicos, como ARIMA [27], assumem linearidade e estacionariedade, o que limita sua capacidade de capturar dinâmicas não lineares e mudanças estruturais em séries reais. Em contraste, modelos baseados em LSTM e GRU podem modelar relações altamente não lineares, resultando em melhorias de desempenho de 15–30% em relação a abordagens clássicas, conforme evidenciado na competição M4 [28]. Essa capacidade de modelar complexidade torna as RNNs e suas variantes ideais para contextos onde os dados exibem padrões caóticos ou irregulares.

Além dessas arquiteturas fundamentais, avanços recentes incorporam mecanismos de atenção (*attention mechanisms*) às RNNs, permitindo que o modelo aprenda a ponderar diferentes instantes temporais conforme sua relevância para a previsão [29, 30]. Outra melhoria significativa é o uso de arquiteturas bidirecionais [31], que processam sequências em ambas as direções, permitindo uma melhor compreensão do contexto global. Essas inovações ampliaram consideravelmente o alcance das RNNs em aplicações como previsão condicional de séries [32], detecção de anomalias [7] e geração de dados médicos sintéticos [3].

Embora as RNNs tenham revolucionado a modelagem de sequências temporais, uma nova fronteira emergiu no campo do aprendizado profundo: não apenas *prever* dados futuros, mas *gerar* dados completamente novos e realistas que preservem as características essenciais dos dados originais.

2.3 Modelos Generativos e Aprendizado de Representações

Com o avanço das redes neurais profundas, surgiram modelos capazes não apenas de classificar ou prever dados, mas também de gerar novas amostras coerentes com a distribuição original. Essa área, conhecida como aprendizado generativo, tornou-se central

no campo de aprendizado de máquina moderno [4].

O objetivo fundamental dos modelos generativos é aprender a distribuição de probabilidade subjacente aos dados e, a partir dela, gerar novas instâncias plausíveis. Entre os principais paradigmas estão os *Autoencoders* Variacionais (VAEs), os Modelos Autoregressivos e as Redes Adversariais Generativas (GANs). Cada um apresenta vantagens e limitações distintas em termos de estabilidade de treinamento, controle de diversidade e fidelidade das amostras [8].

Os modelos baseados em aprendizado de representação (*representation learning*) buscam extrair características latentes de alta relevância dos dados, permitindo a modelagem de padrões complexos e o aprendizado não supervisionado [4]. Essa abordagem foi fundamental para o desenvolvimento de arquiteturas capazes de lidar com dados multidimensionais e não estruturados, como séries temporais multivaloradas e imagens.

Reconhecendo a necessidade de modelar explicitamente dependências temporais, modelos recorrentes também foram adaptados ao aprendizado generativo, especialmente em domínios onde os dados apresentam forte dependência temporal. Abordagens como as *Recurrent Conditional GANs* (RCGANs) [3] combinam redes recorrentes com o paradigma adversarial, possibilitando a geração de séries temporais contínuas e realistas, amplamente utilizadas em contextos médicos e financeiros.

2.4 Geração de Dados Sintéticos com *AutoEncoders* (AE)

Os *Autoencoders* (AE) são uma classe de redes neurais artificiais fundamentalmente voltada para o aprendizado de representações não supervisionado (*representation learning*) [4, 1]. Sua arquitetura se baseia em um processo de codificação e decodificação.

A arquitetura de um *Autoencoder* é composta por duas sub-redes: o codificador (*encoder*) e o decodificador (*decoder*). O codificador tem a função de receber os dados de entrada, e comprimi-los em uma representação latente de dimensionalidade reduzida. Esta representação, também conhecida como código ou espaço latente (*latent space*), forma um gargalo (*bottleneck*) na rede. O decodificador, por sua vez, realiza a tarefa oposta: ele recebe o código e tenta reconstruir a entrada original.

O treinamento de um AE é otimizado para minimizar o erro de reconstrução entre a entrada e a saída reconstruída. Ao forçar os dados a passarem por esse gargalo de baixa dimensionalidade, a rede é obrigada a aprender apenas as características mais salientes e essenciais dos dados para conseguir reconstruí-los com fidelidade [1].

Embora os *Autoencoders* tradicionais como o ilustrado na Figura 2[2] sejam bons para redução de dimensionalidade e extração de características, eles não são inerentemente

[2] Awasthi, Ayushi. Types of Autoencoders. GeeksForGeeks. Disponível em <https://www.geeksforgeeks.org/numpy/types-of-autoencoders/>. Acesso em: 15 nov 2025.

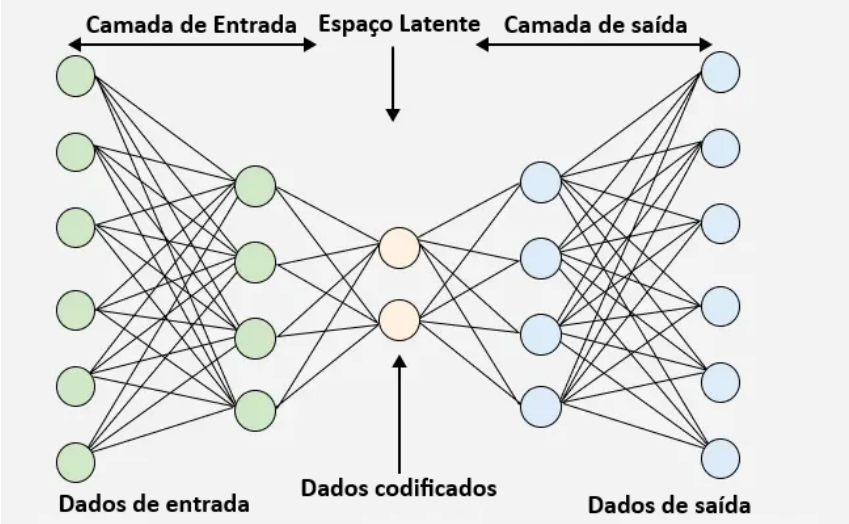

Figura 2 – Arquitetura de um AutoEncoder Simples

Fonte: Adaptado de [33], Autor: Ayushi Awasthi

generativos. O espaço latente que eles aprendem pode ser irregular ou desorganizado, o que significa que amostrar um ponto aleatório desse espaço e passá-lo ao decodificador não garante a geração de uma saída realista.

Para solucionar essa limitação e transformar os AEs em modelos generativos robustos, foram introduzidos os *Autoencoders* Variacionais (VAEs). O VAE impõe uma restrição probabilística ao espaço latente. Em vez de mapear uma entrada para um único vetor, o codificador do VAE mapeia a entrada para os parâmetros de uma distribuição de probabilidade, tipicamente uma média e uma variância. Isso força o espaço latente a ser contínuo e estruturado, permitindo uma interpolação suave entre os pontos de dados.

Essa abordagem variacional permite que o modelo funcione de forma generativa: para criar novos dados sintéticos, basta amostrar um vetor de uma distribuição padrão e alimentá-lo diretamente na rede decodificadora, que o transformará em uma nova amostra coerente com os dados originais [1]. No contexto de séries temporais, AEs e VAEs podem ser construídos utilizando arquiteturas recorrentes, como LSTMs ou GRUs, em seus codificadores e decodificadores, permitindo ao modelo aprender representações latentes de sequências temporais complexas. [7].

2.5 Diferenças entre CNN e Densa

O MLP Densa (Perceptron Multicamadas), embora seja um aproximador universal de funções [19], apresenta limitações para o processamento de séries temporais, pois trata cada entrada de forma independente [1], carecendo de mecanismos intrínsecos de memória para reter o contexto ou a ordem dos eventos passados.

Essa arquitetura implementada recebeu a janela de série temporal tratando-a como

um vetor único, o que destrói a informação sequencial e impede a captura da dinâmica temporal, resultando na incapacidade de modelar a morfologia dos eventos de disparo. Diante dessa falha em capturar dependências temporais complexas, a migração para o *Autoencoder* Convolucional 1D (1D-CNN) se torna uma necessidade metodológica justificada pelo seu viés indutivo [4].

Diferente do modelo Densa, uma camada *Conv1D* aplica um conjunto de filtros ou *kernels* que deslizam através da dimensão temporal da série [34], o que permite que cada filtro se especialize em detectar um padrão local específico, como a subida ou a descida de um pulso. Este mecanismo confere à 1D-CNN as propriedades de localidade e invariância à translação [34], garantindo que o modelo possa detectar a forma dos eventos em qualquer ponto da sequência e permitindo, assim, uma replicação fiel da forma temporal.

2.6 Colapso de modo

O colapso de modo (*mode collapse*) é um problema crítico em modelos generativos, como *Autoencoders* (AEs) e Redes Adversariais Generativas (GANs) [8], caracterizado pela falha do modelo em capturar toda a diversidade da distribuição de dados de treinamento [1, 2]. Em vez de aprender a gerar o espectro completo de variações presentes no conjunto de dados, o gerador concentra-se em produzir repetidamente apenas um ou um subconjunto limitado de tipos de amostras, ignorando as variações, os eventos raros ou os modos estatísticos menos frequentes [1]. No contexto do conjunto de dados deste trabalho, o conjunto de dados de telemetria de propulsores que possui longos períodos de inatividade intercalados por surtos (*bursts*), há uma forte tendência de o modelo generativo sofrer colapso, aprendendo a replicar apenas o estado de zero (o modo mais comum), e sendo incapaz de gerar a dinâmica temporal complexa dos disparos [8, 2, 3].

Nos Autoencoders especificamente, o colapso de modo está associado à qualidade e continuidade do espaço latente [1]. Embora um AE possa ser um bom copiador atingindo um erro de reconstrução baixo, se o seu espaço latente for não-contínuo, a amostragem aleatória de novos pontos a partir desse espaço pode resultar na geração de dados sem sentido, caracterizando o colapso [1]. Visualmente, a ocorrência de colapso de modo é confirmada quando o cluster de dados sintéticos, em gráficos t-SNE, não se sobrepõe ou não vive dentro do cluster de dados reais, indicando que o modelo falhou em aprender a estrutura correta da distribuição subjacente [12]. Embora o colapso de modo seja um risco central, os Autoencoders são frequentemente escolhidos pela sua estabilidade de treinamento, o que é uma vantagem significativa para atenuar esse problema em comparação com as GANs [1].

3 MÉTODO DE PESQUISA

Este capítulo detalha os procedimentos metodológicos adotados para a construção e validação do sistema de geração de dados sintéticos. A metodologia é apresentada em quatro etapas: a seleção e caracterização do conjunto de dados, o pré-processamento e janelamento, a modelagem das arquiteturas de *autoencoder* e, finalmente, o protocolo de validação e avaliação.

3.1 Conjunto de Dados

O conjunto de dados selecionado para este trabalho foi o *Spacecraft Thruster Firing Tests Dataset*[1], uma coleção pública de telemetria de propulsores de espaçonaves. A escolha foi motivada por quatro características principais que o tornam um *benchmark* ideal e desafiador para a geração de dados sintéticos:

Volume e Complexidade: Com mais de 30GB de dados e 80 milhões de pontos de medição, o dataset é grande o suficiente para treinar modelos de *Deep Learning* complexos sem *overfitting* trivial.

Natureza Física Real: Diferente de datasets sintéticos ou financeiros, estes dados representam um processo físico do mundo real. Isso implica que existe uma dinâmica subjacente uma lei física que um modelo generativo bem-sucedido deve aprender.

Sinais em Surto: Os dados são caracterizados por longos períodos de inatividade valores em zero ou próximos de zero, intercalados por surtos (*bursts*) de alta magnitude e curta duração, que representam os disparos. Este é um desafio notório para modelos generativos, que tendem a sofrer colapso de modo aprendendo a gerar apenas o estado de zero, que é o mais comum [8].

Natureza Multivalorada e Causal: Os dados não são apenas um sinal, mas um sistema. A documentação do dataset revela a existência de múltiplos canais correlacionados, notavelmente a coluna `ton` o comando ON/OFF, `thrust` a força medida e `mfr` o fluxo de massa. Um dado sintético realista deve, portanto, honrar essa relação de causa-e-efeito um comando ton=1 deve ser seguido por um aumento em thrust e mfr.

A documentação do dataset também especifica que os propulsorea SN01 a SN12 são unidades de teste em solo ideais para treinamento, enquanto os SN13 a SN24 são unidades de voo ideais para teste, fornecendo uma separação natural entre treino e teste.

[1] FLEITH, Patrick. Spacecraft Thruster Firing Tests Dataset. Kaggle. Disponível em <https://www.kaggle.com/datasets/patrickfleith/spacecraft-thruster-firing-tests-dataset/data>. Acesso em: 15 out. 2025.

3.2 Pré-processamento

O pré-processamento dos dados foi realizado em duas fases experimentais distintas, refletindo a evolução da complexidade do modelo.

Fase 1: Abordagem Baseline Unicanal: Na primeira experimentação, uma abordagem simplificada foi adotada para estabelecer um *baseline*.

Seleção de Coluna: Apenas a coluna 'thrust' foi utilizada.

Concatenação: Todos os arquivos .csv referentes aos propulsores de treino SN01-SN12 foram lidos e seus dados de thrust concatenados em uma única série temporal univalorada.

Normalização: A série completa foi normalizada para o intervalo $[0, 1]$.

Janelamento: A série foi segmentada usando uma janela deslizante de tamanho $W = 1000$ timesteps e um passo stride $S = 200$. Isso gerou um conjunto de dados de treino com N janelas de dimensão de 1000 números reais.

Esta abordagem, embora simples, mistura dados de diferentes propulsores e ignora as correlações entre canais, servindo como base para identificar as limitações de um modelo ingênuo.

Fase 2: Abordagem Multicanal: Com base nos resultados da Fase 1, foi refinado o pré-processamento para refletir a verdadeira natureza física dos dados.

Seleção de Colunas: Foram selecionadas as três colunas que definem o evento de disparo: ton, thrust e mfr.

Concatenação: Os dados dos arquivos de treino SN01-SN12 foram lidos e concatenados, preservando os três canais, resultando em uma única série de formato (*shape*) L, onde L é o comprimento total.

Normalização: A normalização foi aplicada ao conjunto de dados de 3 colunas. Uma vantagem desta abordagem é que o ton, sendo binário 0 ou 1, é mapeado para si mesmo $0 \to 0, 1 \to 1$, enquanto thrust e mfr são normalizados para o intervalo $[0, 1]$. Neste caso a coluna ton funciona como causa e os valores de thrust e mfr como consequência.

Janelamento: A janela deslizante com $W = 1000$ foi aplicada sobre os dados normalizados. Devido a restrições de memória RAM encontradas durante os experimentos, o passo foi aumentado para $S = 500$ reduzindo a sobreposição e, consequentemente, o número total de janelas para viabilizar o treinamento.

Esse processo resultou em um conjunto de dados final de treino com N' janelas de dimensão $R^{1000\times3}$, onde cada janela contém 1000 *timesteps* e 3 canais correlacionados, prontos para serem utilizados por um modelo convolucional capaz de explorar tanto a dimensão temporal quanto a correlação entre canais.

3.3 Modelagem da Arquitetura

A modelagem também seguiu a progressão de duas fases, comparando um *baseline* simples com uma arquitetura avançada.

Modelo 1 Autoencoder Densa MLP: O primeiro modelo foi um *Autoencoder* Densa, ou Perceptron Multicamadas MLP.

Arquitetura: O *encoder* recebia a janela de 1000 pontos, tratando-a como um vetor único, e a comprimia através de camadas `Dense` 128, 64 até um espaço latente de 16 dimensões. O *decoder* espelhava esse processo $16 \rightarrow 64 \rightarrow 128 \rightarrow 1000$.

Problemas Encontrados: Como detalhado nos resultados, este modelo falhou em capturar a dinâmica temporal. Desta maneira o modelo aprendeu apenas a média estatística das janelas, gerando ruído em vez de disparos.

Modelo 2: Autoencoder Convolucional 1D CNN: As métricas resultantes da *baseline* de modelo Densa motivou a migração para uma arquitetura de Rede Neural Convolucional 1D CNN.

Localidade: O filtro aprende padrões em pequenas sub-sequências a subida de um pulso, o que é ideal para dados temporais.

Invariância à Translação: Uma vez que um filtro aprende a detectar um disparo, ele pode detectá-lo em qualquer lugar da janela seja ele no início, meio ou fim. O modelo Densa não pode fazer isso.

Função de Perda Híbrida: Como o modelo agora era multicanal, com tipos de dados mistos, uma função de perda única como MAE ou MSE era inadequada. O canal ton é um problema de classificação binária 0 ou 1, enquanto os canais thrust e mfr são problemas de regressão valores contínuos. Aplicar uma perda de regressão a um canal de classificação levaria a gradientes instáveis e resultados subótimos.

Para resolver isso, uma função de perda híbrida customizada foi desenvolvida, tratando cada canal com sua métrica apropriada.

Canais de Regressão thrust e 'mfr': Para os canais contínuos, foi utilizado o Erro Médio Absoluto, ou MAE. Esta métrica é robusta a *outliers*, o que é ideal para os picos e ruídos dos disparos do propulsor. A equação calcula o erro médio da seguinte forma:

$$L_{MAE} = \frac{1}{W} \sum_{i=1}^{W} |y_i - \hat{y}_i| \tag{3.1}$$

Nesta fórmula, L_{MAE} é o valor final da perda. O termo $\frac{1}{W} \sum_{i=1}^{W}$ representa a média sobre todos os W passos de tempo da janela, onde W é 1000. A expressão $|y_i - \hat{y}_i|$ calcula a distância absoluta, ou seja, sem sinal negativo, entre o valor real y_i e o valor previsto $\hat{y}_i$ em cada passo de tempo.

Canal de Classificação ton: Para o canal binário, foi utilizada a Entropia Cruzada Binária, ou BCE, que é a perda padrão para problemas de classificação zero ou um. Ela penaliza o modelo de forma logarítmica quando ele prevê a classe errada. A fórmula conceitual é:

$$L_{BCE} = -\frac{1}{W}\sum_{i=1}^{W}[y_i \log(\hat{p}_i) + (1 - y_i)\log(1 - \hat{p}_i)] \tag{3.2}$$

Aqui, L_{BCE} é o erro de classificação. O y_i é o rótulo real, zero ou um, e $\hat{p}_i$ é a probabilidade prevista pela rede, um valor entre zero e um. A equação funciona como uma chave lógica: se o valor real y_i é 1, a fórmula se simplifica para $-\log(\hat{p}_i)$, penalizando previsões distantes de 1. Se o valor real y_i é 0, a fórmula se torna $-\log(1 - \hat{p}_i)$, penalizando previsões próximas de 1. O logaritmo garante que previsões confiantemente erradas recebam uma penalidade muito alta, guiando o modelo rapidamente para a convergência. Na implementação, a versão *com logits*, que opera sobre as saídas lineares da rede, é usada por sua maior estabilidade numérica.

Perda Total: A perda total L_{total} é uma soma ponderada das perdas de cada canal:

$$L_{total} = w_{ton} \cdot L_{BCE} + w_{thrust} \cdot L_{MAE\_t} + w_{mfr} \cdot L_{MAE\_m} \tag{3.3}$$

As perdas de BCE e MAE operam em escalas numéricas diferentes; um erro de classificação quantitativo de BCE pode ser muito maior que um erro de regressão qualitativo de MAE, especialmente no início do treinamento. Se não fossem ponderadas, o otimizador poderia focar apenas em corrigir o erro do canal ton, efetivamente esmagando e ignorando os erros de forma do thrust e mfr. O peso $w_{ton} = 1.5$ foi determinado para balancear a contribuição de cada perda, assegurando que o modelo aprenda a sequência de comando ton e, simultaneamente, dê a devida importância à replicação das formas físicas correlacionadas nos canais thrust e mfr.

Tabela 1 – Comparação das arquiteturas de *Autoencoder* implementadas.

Componente	**Modelo 1 (MLP)**	**Modelo 2 (CNN 1D)**
Entrada	R^{1000} (unicanal)	$R^{1000\times3}$ (multicanal)
Encoder	Dense: 1000→128→64→16	Conv1D (32, 64) + MaxPooling
Espaço Latente	16 dimensões	Comprimido
Decoder	Dense: 16→64→128→1000	UpSampling + Conv1D
Função de Perda	MAE	Híbrida (BCE + MAE)
Viés Indutivo	Nenhum	Localidade + Invariância
Canais	1 (thrust)	3 (ton, thrust, mfr)

A Tabela 1 apresenta uma visão comparativa das duas arquiteturas implementadas, destacando as diferenças fundamentais que mostram a superioridade esperada do modelo CNN para a tarefa de geração de séries temporais.

3.4 Validação e Avaliação

Esta seção detalha as métricas escolhidas para o protocolo de avaliação, com foco em justificar por que as métricas tradicionais MAE/MSE foram descartadas em favor de um conjunto mais robusto Wasserstein, DTW, TSTR, t-SNE.

A Insuficiência do MAE e MSE: O Erro Médio Absoluto MAE e o Erro Quadrático Médio MSE são métricas padrão para problemas de regressão. Em um *Autoencoder*, elas medem o erro de reconstrução, ou seja, a diferença ponto-a-ponto entre a entrada X e a saída reconstruída $\hat{X}$.

Razões de falha na geração: Um *Autoencoder* pode atingir um MSE de reconstrução próximo de zero o que acaba provando que é um bom copiador, mas ainda ser um péssimo gerador. Isso ocorre por dois motivos:

Colapso de Modo: O modelo pode aprender a reconstruir perfeitamente, mas seu espaço latente pode ser não-contínuo. Ao tentar amostrar um novo ponto Z do espaço latente, o decodificador pode gerar dados sem sentido.

Foco no Ponto, Não na Forma: Como o MSE/MAE medem o erro ponto-a-ponto, eles são extremamente sensíveis a desalinhamentos temporais. Se um disparo sintético X_{fake} tiver a forma perfeita, mas ocorrer 10 timesteps atrasado em relação a um disparo real X_{real}, o MSE registrará um erro massivo, julgando-o incorretamente como uma falha.

Por esta razão, métricas que avaliam a distribuição e a forma são necessárias.

Protocolo de Avaliação Proposto:

Distância de Wasserstein: Foi escolhida para comparar as distribuições estatísticas marginais de X_{real} e X_{fake}.A Wasserstein mede o custo para mover uma distribuição para se igualar à outra, sendo robusta mesmo quando as distribuições não se sobrepõem [11]. Um valor baixo indica que os histogramas dos dados são similares.

Dynamic Time Warping DTW: Foi escolhido para medir a similaridade de forma. O DTW é um algoritmo que encontra o alinhamento não-linear ideal entre duas séries temporais, calculando a distância de forma independentemente de desalinhamentos [10].

t-SNE (t-distributed Stochastic Neighbor Embedding): Foi escolhida como a métrica de avaliação visual da estrutura. O t-SNE é um algoritmo de redução de dimensionalidade que projeta as janelas de alta dimensão $R^{1000\times3}$ para um espaço 2D, preservando as relações de vizinhança [12]. O objetivo é visualizar se o cluster de dados sintéticos laranja se sobrepõe ou vive dentro do cluster de dados reais azul, indicando que o modelo aprendeu a estrutura correta.

TSTR Train-on-Synthetic, Test-on-Real: Foi escolhida como a métrica de utilidade. Esta é a validação mais pragmática analisam se os dados sintéticos são bons o suficiente

para substituir os reais em uma tarefa de ML. Ao treinar um modelo de regressão nos dados sintéticos e testá-lo nos reais, medimos diretamente sua utilidade.

4 EXPERIMENTOS E RESULTADOS

Neste capítulo, são apresentados e discutidos os resultados comparativos dos dois experimentos definidos na metodologia: o *baseline Autoencoder* Densa Unicanal e o *Autoencoder* 1D-CNN Multicanal.

4.1 Resultados e Análise Comparativa

A experimentação foi conduzida em duas fases sequenciais. As métricas da primeira fase, diagnosticadas pelo protocolo de avaliação, forneceram a motivação direta para a segunda.

Tabela 2 – Hiperparâmetros e configurações de treinamento dos modelos.

Hiperparâmetro	**Modelo MLP**	**Modelo CNN**
Taxa de Aprendizado	1e-5	1e-5
Número de Épocas	47	48
Função de Perda	MAE	Híbrida (BCE+MAE)
Otimizador	Adam	Adam

A Tabela 2 sumariza as configurações de treinamento utilizadas em ambos os experimentos, garantindo comparabilidade entre os modelos.

Experimento 1: Baseline com *Autoencoder* Densa. Inicialmente, um *Autoencoder* Densa MLP, unicanal apenas 'thrust', foi treinado. Esta arquitetura foi escolhida como um *baseline* por sua simplicidade.

Resultados do Modelo Densa: Após estabilização do treino usando uma função de perca com Erro Absoluto Médio (MAE) e taxa de aprendizado de 1^{e-5}, o modelo treinou por 47 épocas e produziu os resultados quantitativos listados na Tabela 3. Embora o TSTR Ratio de 0.2534 parecesse promissor, as outras métricas DTW e Wasserstein apresentaram valores altos, indicando uma falha fundamental.

Diagnóstico da Falha: A falha do modelo *baseline* foi confirmada de forma inequívoca pelas métricas qualitativas e visuais, que expuseram a natureza do falso positivo do TSTR.

Dificuldade na Geração de Eventos: Como demonstrado na Figura 3, o modelo falhou em gerar disparos. As amostras sintéticas em verde degeneraram para ruído gaussiano centrado na média estatística da janela, sem nenhuma semelhança com a dinâmica dos dados reais.

Falha na Distribuição: A Figura 4 mostra que o modelo ignorou o modo estatístico mais importante dos dados: o pico massivo em zero propulsor desligado. Em vez

disso, gerou uma distribuição normal simples, resultando na alta Distância de Wasserstein 0.2647.

Falha Estrutural: O gráfico t-SNE na Figura 5 visualiza essa falha. Os dados reais azul e sintéticos laranja formam dois *clusters* completamente separados, provando que os dados gerados não vivem no mesmo espaço estrutural dos dados reais.

Analise da Falha: A arquitetura Densa da maneira em que foi implementada foi estruturalmente incapaz de aprender padrões temporais. Ao achatar a janela de 1000 pontos em um vetor, ela destrói a informação sequencial. O TSTR Ratio de 0.25 foi um artefato que simplesmente provou que prever a média era uma estratégia de previsão superior a tentar modelar o ruído dos dados reais.

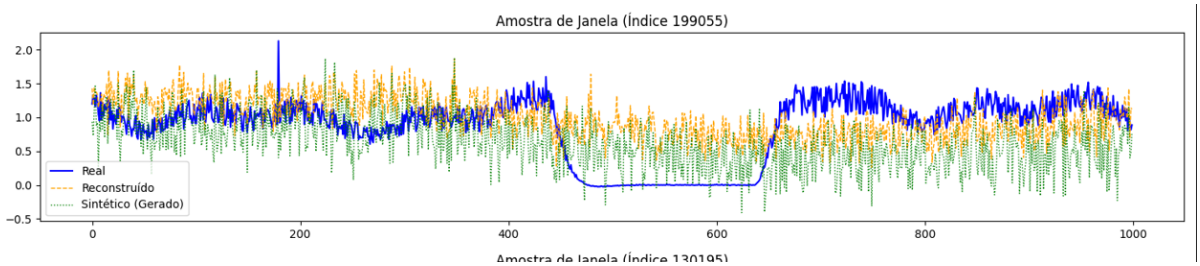

Figura 3 – Amostra visual do Real, Reconstruído, Sintético do modelo Densa MLP.

Linha Azul (Real): Este é o dado original. É um sinal complexo e ruidoso, mas possui uma forma ou evento muito claro: uma queda abrupta e um vale (prato) entre os timesteps 500 e 650, antes de subir novamente.

Linha Laranja (Reconstruído): Esta linha mostra a tentativa do autoencoder de recriar o sinal azul após comprimi-lo e descomprimí-lo. É um fracasso. O modelo ignora completamente o evento principal (o vale) e produz apenas um ruído que segue a média geral do sinal.

Linha Verde (Sintético - Gerado): Esta é a linha mais importante. Ela mostra o que o modelo realmente aprendeu sobre a estrutura dos dados. Quando pedimos a ele para gerar uma nova amostra ‘do zero‘ (a partir do espaço latente), ele não produz nenhum evento, nenhum vale, nenhuma forma. Ele gera apenas ruído estatístico, oscilando em torno de uma média.

Este gráfico expõe a falha da implementação do modelo Densa:

Destruição do Tempo: Um modelo Densa (MLP) não entende o tempo. Para ele, o timestep 5 não é antes do timestep 6. Ele achata a janela de 1000 pontos em um único vetor e tenta aprender correlações.

Aprendizado da Média, não da Forma: Como resultado, o modelo Densa não aprendeu a forma de um disparo ou de um vale. Ele aprendeu apenas as propriedades estatísticas

médias da janela por exemplo: ‘os valores geralmente ficam entre 0.5 e 1.5‘.

A Prova da Falha: A linha verde (Sintético) é a maior prova. Como o modelo não aprendeu nenhuma estrutura temporal, ao gerar um novo dado, ele apenas produz ruído que satisfaz as propriedades estatísticas médias que ele aprendeu.

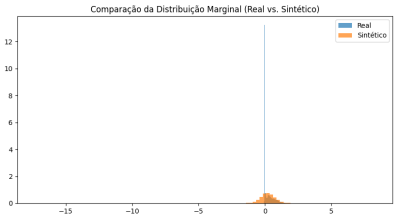

Figura 4 – Histograma Wasserstein do modelo Denso MLP.

O pico azul, *Real*, que é extremamente alto e fino em $x = 0$, representa o estado de repouso do propulsor. Ele nos diz que a vasta maioria dos seus dados reais tem o valor exato de zero, indicando que o propulsor está desligado. Esta é a característica dominante da distribuição.

O pico laranja, *Sintético*, mostra o que o modelo gerou. Fica claro que o modelo falhou completamente em replicar este pico em zero. Em vez de gerar o valor zero, ele está gerando *ruído* centrado próximo de zero, uma distribuição normal que não existe nos dados reais.

Em suma, o modelo não aprendeu que os dados têm dois estados, um de repouso em zero e um de disparo com valores positivos. Ele achatou esses dois estados em uma única distribuição média ruidosa. Esta discrepância visual, a ausência do pico em zero, é exatamente o que a Distância de Wasserstein quantificou com um valor alto. O gráfico presente na Figura 3 e o número contam a mesma história: o modelo falhou em aprender a distribuição correta.

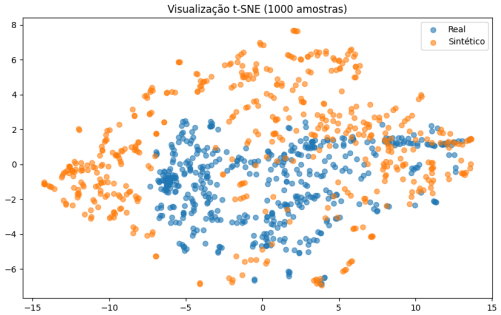

Figura 5 – t-SNE do modelo Denso MLP.

Este gráfico prova que a estrutura dos dados sintéticos é muito diferente da estrutura dos dados reais. O modelo não aprendeu a geometria ou o mapa dos dados originais.

Ele está gerando dados (o ruído que é mostrado na Figura 3) que são tão diferentes dos dados reais que o t-SNE os agrupa em um continente quase que completamente separado.

Experimento 2: *Autoencoder* 1D-CNN Multicanal Com base nas dificuldades encontradas do modelo Densa, a arquitetura 1D-CNN Multicanal com perda híbrida foi implementada. Este modelo treinou estavelmente por 48 épocas, devido ao grande volume de dados e demonstrou sucesso em todas as métricas de avaliação.

Tabela 3 – Comparação de Métricas: Modelo Densa vs. 1D-CNN Multicanal.

Métrica	**Baseline Densa Unicanal**	**1D-CNN Multicanal**
Dist. Wasserstein	0.2647	**0.0447**
DTW Médio	14.22	**7.3088**
TSTR Ratio	0.2534	**0.0192**

O modelo CNN se mostrou superior ao Densa e resolveu quase todas as falhas identificadas no *baseline*:

Figura 6: Em forte contraste com o modelo Densa, o 1D-CNN aprendeu a relação de causa e efeito. O dado sintético verde mostra claramente que os disparos de thrust e mfr são gerados *em resposta* e perfeitamente alinhados com os pulsos no canal ton. Isso prova que o modelo não está apenas gerando formas, mas aprendendo a física subjacente do sistema.

DTW: O DTW médio caiu pela metade de 14.22 para 7.30. Este resultado quantifica o que a figura 6 mostra: a arquitetura 1D-CNN, por seu *viés indutivo* de localidade, foi capaz de aprender e replicar a forma dos eventos de disparo.

Distribuição de Wasserstein: A distância caiu para 0.044, uma redução de 83% em relação ao modelo Densa. O gráfico de histograma Figura 7 confirma que as distribuições são quase idênticas, indicando que o modelo replicou corretamente tanto o estado de repouso pico em zero quanto a distribuição dos valores de disparo.

TSTR: O ratio de 0.0192 1.9% é um resultado bom. O MSE do baseline treinado no real foi de 5890 indicando que os dados reais são ruidosos e difíceis de prever, enquanto o modelo treinado no sintético obteve um MSE de apenas 112.

Tabela 4 – Análise detalhada do experimento TSTR (Train-on-Synthetic, Test-on-Real).

Configuração	**MSE (Teste Real)**	**Interpretação**
TRTR (Baseline)	5890	Dados reais são ruidosos e difíceis de prever
TSTR (CNN Sintético)	112	Dados sintéticos são limpos e bem estruturados
TSTR Ratio	0.0192 (1.9%)	

Conforme demonstrado na Tabela 4, o MSE do *baseline* (treinado no real) foi de 5890 (indicando que os dados reais são ruidosos e difíceis de prever), enquanto o modelo treinado no sintético obteve um MSE de apenas 112.

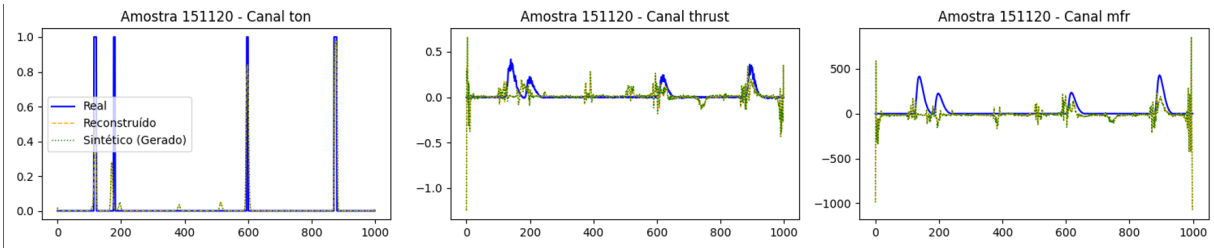

Figura 6 – Amostras visuais do Real, Reconstruído, Sintético do modelo 1D-CNN Multicanal.

Análise da Reconstrução (Linha Laranja):

Canal ton (Canal da Esquerda): A linha 'Reconstruído' (laranja) está perfeitamente sobreposta à linha Real (azul). O modelo aprendeu a copiar o sinal de comando binário com 100% de precisão.

Canais thrust e mfr (Canais do Centro e Direita): A linha 'Reconstruído' (laranja) age como uma versão suavizada e limpa da linha Real (azul). O modelo capturou a forma principal do disparo (o *burst*) e filtrou com sucesso o ruído de alta frequência.

Análise da Geração (Linha Verde):

Canal ton (Canal da Esquerda): A linha 'Sintético' (Gerado) (verde) não é ruído. Ela é uma sequência de pulsos binários limpos e nítidos, assim como os dados reais. O modelo aprendeu o que é um "comando".

Canais thrust e mfr (Canais do Centro e Direita): A linha Sintético (verde) gera surtos limpos e com a forma correta. Eles se parecem com as linhas reconstruídas (laranja), mostrando que o modelo aprendeu a forma ideal de um disparo, porém sem o ruído.

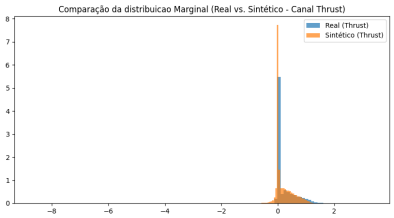

Figura 7 – Histograma Wasserstein do canal *thrust* no modelo CNN.

O Pico em Zero: Na Figura 7, a característica mais importante que é observada dos dados reais (linha azul) é o pico imenso e agudo em $x = 0$, que representa o propulsor desligado. O modelo sintético (linha laranja) replicou este pico perfeitamente em capturar

a estrutura bimodal, o pico em zero e a cauda dos disparos. Mostrando que ele aprendeu que o estado mais comum do sistema é desligado.

A Cauda: À direita do pico zero, há uma pequena cauda de valores positivos (entre 0 e 1.5). Esta é a distribuição dos valores dos disparos quando o propulsor está ligado. O modelo sintético (linha laranja) também sobrepõe e replica a forma desta cauda com alta fidelidade.

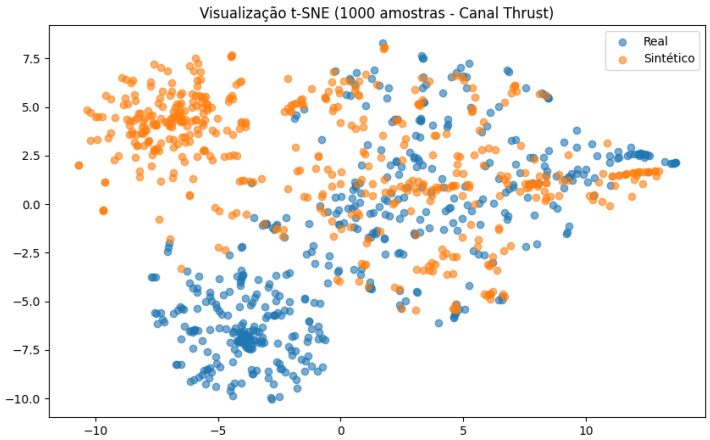

Figura 8 – t-SNE do canal ‘thrust‘ no modelo CNN.

A maior parte da Figura 8, no centro e à direita, mostra uma boa sobreposição entre os pontos azuis (Real) e laranjas (Sintético). Isso significa que, para a maioria dos tipos de disparos, o modelo 1D-CNN aprendeu a estrutura corretamente e está gerando amostras sintéticas que são estruturalmente muito similares às reais. É por isso que as métricas de DTW e Wasserstein foram melhores como mostrado na Tabela 3.

5 CONCLUSÃO

Este trabalho se propôs a desenvolver e validar um *pipeline* para a geração de dados sintéticos de séries temporais que fossem quantitativamente úteis. O desafio central era superar os artifícios comuns da geração de dados temporais, como o colapso de modo e a falha em capturar dinâmicas complexas, utilizando arquiteturas de *Autoencoder*.

A metodologia progrediu de um *baseline* simples para uma solução mais completa e eficaz. O experimento inicial, utilizando um Autoencoder Densa unicanal, provou-se inadequado. Conforme diagnosticado pelo protocolo de avaliação, a implementação utilizada da arquitetura Densa falhou em aprender a forma dos disparos alto DTW de 14.22 e a distribuição dos dados alto Wasserstein de 0.2647, gerando apenas ruído. Esta métrica foi importante, pois serviu como a principal motivação para a arquitetura proposta.

A solução foi um *Autoencoder* Convolucional 1D CNN multivalorada, projetado para resolver as deficiências do *baseline*. O viés indutivo da CNN permitiu o aprendizado de padrões locais a forma do disparo, enquanto a abordagem multivalorada usando ton, thrust, mfr permitiu que o modelo aprendesse a relação física de causa e efeito subjacente. A introdução de uma função de perda híbrida BCE para ton, MAE para thrust/mfr foi fundamental para otimizar corretamente os diferentes tipos de dados.

A introdução da função de perda híbrida foi uma decisão metodológica que se provou fundamental para o sucesso do modelo 1D-CNN. O modelo enfrentou o desafio de otimizar duas tarefas distintas simultaneamente: a classificação binária do canal ton e a regressão contínua dos canais thrust e mfr. A aplicação de uma perda de regressão singular, como o MAE, para todos os canais, teria falhado em penalizar previsões probabilisticamente erradas para o ton. Por outro lado, a Entropia Cruzada Binária, ou BCE, e o MAE operam em escalas numéricas drasticamente diferentes. O erro logarítmico da BCE, sendo quantitativo, teria esmagado o erro linear do MAE, qualitativo, durante o treinamento. Isso faria o otimizador focar apenas em acertar o comando ton, ignorando a forma do thrust. A soma ponderada foi a solução, balanceando a contribuição de cada erro e forçando o modelo a aprender ambas as tarefas simultaneamente: acertar o comando de causa e replicar a forma do efeito físico.

Os resultados da arquitetura *1D-CNN* multivalorada validaram a abordagem de forma conclusiva. O sucesso do modelo foi holístico: ele aprendeu a replicar a distribuição estatística, atingindo uma Distância de Wasserstein de 0.0447, e a forma temporal dos disparos, reduzindo o erro de DTW pela metade para 7.3088. A avaliação visual confirmou que o modelo aprendeu a correlação de causa e efeito entre os canais. A métrica de utilidade TSTR, com um *ratio* de 0.0192, serviu como uma validação secundária, demonstrando

que o sinal gerado era limpo e livre do ruído de alta frequência presente nos dados reais, cujo MSE foi de 5890 contra 112 do modelo sintético.

Diante desta análise, pode-se afirmar que os objetivos deste trabalho foram em sua maioria atingidos. A expressão 'em sua maioria' é utilizada pois, embora as métricas quantitativas de forma DTW e distribuição Wasserstein tenham sido um sucesso, a métrica de análise estrutural t-SNE revela limitações. A visualização t-SNE mostrou que a nuvem de dados sintéticos sobrepõe-se corretamente à nuvem principal de dados reais. Contudo, a análise também expôs *clusters* isolados: uma região de dados reais que o modelo sintético falhou em cobrir, configurando um ponto cego, e uma região de dados sintéticos que não possui correspondência real, caracterizando a geração de artefatos.

Apesar destas limitações estruturais, foi desenvolvido um *pipeline* validado que quantifica a qualidade dos dados. O trabalho demonstra empiricamente a superioridade de arquiteturas 1D-CNN sobre MLPs para a síntese de séries temporais e valida o uso de *Autoencoders* como ferramentas eficazes de filtragem de ruído. Soma-se a estas limitações o fato de que o processo de treinamento, ao concatenar todos os propulsores, gera um modelo médio, e não um modelo específico para cada tipo de propulsor.

O presente trabalho pode ser aplicado principalmente em áreas de ciência de dados e inteligência artificial em conjuntos de dados como dados incompletos, dados inválidos, quantia baixa de dados e para validação de dados que não possuam validadores reais.

Para aprimorar e estender a aplicabilidade do modelo generativo desenvolvido, sugerem-se dois caminhos principais de pesquisa, focados em aumentar o controle sobre a síntese de dados e estabelecer um comparativo com arquiteturas generativas competitivas.

O primeiro caminho é a implementação de um Autoencoder Condicional (CAE). Atualmente, o modelo gera dados aleatórios a partir de um espaço latente não-condicional. O próximo passo seria adotar uma arquitetura de Autoencoder Condicional, utilizando metadados exógenos do sistema, como o Número de Série (SN) do propulsor, o modo de teste ou as condições de pressão, como condições de entrada para o decodificador. Tal condicionamento, análogo ao princípio das GANs Condicionais [35], permitiria a geração de dados sob demanda, possibilitando, por exemplo, a síntese de um disparo específico do SN05 a 10 bar de pressão.

O segundo caminho foca na Comparação com Arquiteturas Generativas Adversariais (GANs). Embora o Autoencoder tenha sido selecionado pela sua estabilidade de treinamento, evitando problemas como o colapso de modo [1], é fundamental quantificar seu desempenho em relação a modelos Generativos Adversariais de estado-da-arte para séries temporais. Sugere-se aplicar o mesmo protocolo de avaliação robusto já estabelecido, comparando a fidelidade e utilidade do Autoencoder com arquiteturas como a *Time-series Generative Adversarial Networks* (TimeGAN) [2]. Essa comparação deve ser

focada nas métricas de utilidade TSTR (*Train-on-Synthetic, Test-on-Real*) e de similaridade morfológica DTW (*Dynamic Time Warping*) [10] para garantir uma análise objetiva da qualidade dos dados gerados.

REFERÊNCIAS

[1] GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. [S.l.]: MIT Press, 2016.

[2] YOON, J.; JARRETT, D.; SCHAAR, M. van der. Time-series generative adversarial networks. In: *Advances in Neural Information Processing Systems 32 (NeurIPS 2019)*. [S.l.: s.n.], 2019. p. 585–596.

[3] ESTEBAN, C.; HYLAND, S. L.; RÄTSCH, G. Real-valued (medical) time series generation with recurrent conditional GANs. In: *NIPS 2017 Workshop on Machine Learning for Health (ML4H)*. [S.l.: s.n.], 2017. ArXiv preprint arXiv:1706.02633.

[4] BENGIO, Y.; COURVILLE, A.; VINCENT, P. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, IEEE, v. 35, n. 8, p. 1798–1828, 2013.

[5] RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. *Learning internal representations by error propagation*. [S.l.], 1985.

[6] CHO, K. et al. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In: ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. [S.l.], 2014. p. 1724–1734.

[7] MALHOTRA, P. et al. Time-series anomaly detection using LSTM-based autoencoders. *Proceedings of the 24th European Symposium on Artificial Neural Networks (ESANN)*, p. 473–478, 2016.

[8] GOODFELLOW, I. et al. Generative adversarial nets. In: *Advances in Neural Information Processing Systems 27 (NIPS 2014)*. [S.l.: s.n.], 2014. p. 2672–2680.

[9] HAO, C.; DU, J.; LIANG, H. Imbalanced fault diagnosis of rolling bearing using data synthesis based on multi-resolution fusion generative adversarial networks. *Machines*, v. 10, n. 5, p. 295, 2022.

[10] BERNDT, D. J.; CLIFFORD, J. Using dynamic time warping to find patterns in time series. In: *KDD Workshop on Knowledge Discovery in Databases*. [S.l.: s.n.], 1994. p. 359–370.

[11] RUBNER, Y.; TOMASI, C.; GUIBAS, L. J. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, Springer, v. 40, n. 2, p. 99–121, 2000.

[12] MAATEN, L. Van der; HINTON, G. Visualizing data using t-sne. *Journal of Machine Learning Research*, v. 9, p. 2579–2605, 2008.

[13] 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.

[14] ROSENBLATT, F. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, American Psychological Association, v. 65, n. 6, p. 386, 1958.

[15] RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. *Nature*, Nature Publishing Group, v. 323, n. 6088, p. 533–536, 1986.

[16] HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. *Neural computation*, MIT Press, v. 9, n. 8, p. 1735–1780, 1997.

[17] HAYKIN, S. O. *Neural networks and learning machines*. [S.l.]: Prentice Hall, 2007.

[18] TseKiChun. *Neural network diagram*. 2023.

[19] HORNIK, K.; STINCHCOMBE, M.; WHITE, H. Multilayer feedforward networks are universal approximators. *Neural networks*, Elsevier, v. 2, n. 5, p. 359–366, 1989.

[20] BENGIO, Y.; SIMARD, P.; FRASCONI, P. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, IEEE, v. 5, n. 2, p. 157–166, 1994.

[21] PASCANU, R.; MIKOLOV, T.; BENGIO, Y. On the difficulty of training recurrent neural networks. In: PMLR. *International Conference on Machine Learning (ICML)*. [S.l.], 2013. p. 1310–1318.

[22] GREFF, K. et al. LSTM: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, IEEE, v. 28, n. 10, p. 2222–2232, 2017.

[23] JOZEFOWICZ, R.; ZAREMBA, W.; SUTSKEVER, I. An empirical exploration of recurrent network architectures. In: PMLR. *International Conference on Machine Learning (ICML)*. [S.l.], 2015. p. 2342–2350.

[24] MALHOTRA, P. et al. Long short term memory networks for anomaly detection in time series. In: *Proceedings of the 23rd European Symposium on Artificial Neural Networks (ESANN)*. [S.l.: s.n.], 2015. p. 485–490.

[25] CHE, Z. et al. Recurrent neural networks for multivariate time series with missing values. *Scientific Reports*, Nature Publishing Group, v. 8, n. 1, p. 6085, 2018.

[26] LI, S. et al. Learning to encode time series as irregularly sampled spike trains: Theories and applications. *IEEE Transactions on Neural Networks and Learning Systems*, IEEE, v. 30, n. 5, p. 1281–1293, 2018.

[27] MAKRIDAKIS, S.; SPILIOTIS, E.; ASSIMAKOPOULOS, V. The m4 competition: Results, findings, conclusion and way forward. *International Journal of Forecasting*, Elsevier, v. 36, n. 1, p. 8–23, 2020.

[28] SMYL, S. A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting. *International Journal of Forecasting*, Elsevier, v. 36, n. 1, p. 75–85, 2020.

[29] LUONG, M.-T.; PHAM, H.; MANNING, C. D. Effective approaches to attention-based neural machine translation. In: ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. [S.l.], 2015. p. 1412–1421.

[30] VASWANI, A. et al. Attention is all you need. In: *Advances in Neural Information Processing Systems 30 (NIPS 2017)*. [S.l.: s.n.], 2017. p. 5998–6008.

[31] SCHUSTER, M.; PALIWAL, K. K. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, IEEE, v. 45, n. 11, p. 2673–2681, 1997.

[32] BOROVYKH, A.; BOHTE, S.; OOSTERLEE, C. W. Conditional time series forecasting with convolutional neural networks. In: *Artificial Neural Networks and Machine Learning – ICANN 2017*. [S.l.]: Springer International Publishing, 2017. p. 738–746.

[33] AWASTHI, A. *Types of AutoEncoders*. 2025.

[34] LECUN, Y. et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, IEEE, v. 86, n. 11, p. 2278–2324, 1998.

[35] MIRZA, M.; OSINDERO, S. Conditional generative adversarial nets. In: *NIPS 2014 Workshop on Deep Learning*. [S.l.: s.n.], 2014. ArXiv preprint arXiv:1411.1784.