

Desenvolver um modelo gerador de séries temporais multivariadas para um compressor industrial offshore baseado em RNN e VAEs.

Rodrigo Petrus Domingues

CPE727 – Deep Learning

8 de dezembro de 2025

Sumário

- 1 Motivação e Objetivos
- 2 Base de dados e pré-processamento
- 3 Formulação do Problema
- 4 Arquiteturas e Variações
- 5 Configuração Experimental
- 6 Resultados
- 7 Discussão e Conclusões

Motivação e Objetivos do trabalho

Motivação:

- Compressores industriais são ativos críticos (segurança, disponibilidade, custo).
- As séries temporais multivariadas de sensores refletem o estado operacional.
- Modelos gerativos podem simular cenários operacionais variados.
- Gêmeos digitais e simulações realistas auxiliam na manutenção preditiva

Objetivos do trabalho:

- Desenvolver **modelos geradores** de séries temporais multivariadas para um compressor industrial offshore.
- Aprender, de forma não supervisionada, a distribuição dos sinais de processo em janelas temporais.
- Gerar novas trajetórias plausíveis para:
 - simulação de cenários,
 - testes de algoritmos de monitoramento,
 - análise de variabilidade operacional.

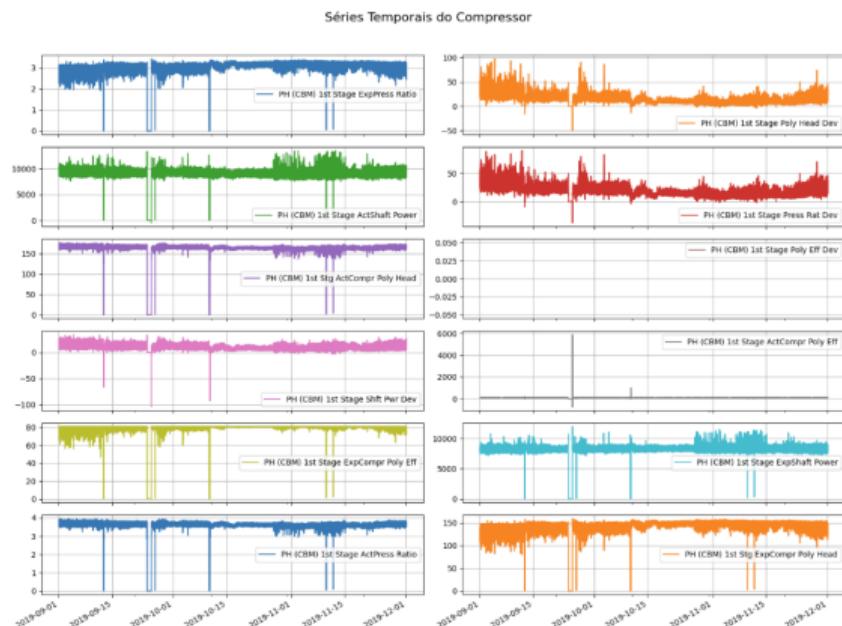
Comparar arquiteturas:

- Modelos DEEP, LSTM, GRU, BiLSTM e BiGRU com GRU fuser, todos com otimização Adam;
- Variações de treinamento (warmup, scheduler, RAdam, AdamW, variational dropout, difusão e layernorm).

Base de dados e pré-processamento

Base de dados

Cognite — sinais reais de sensores de um compressor industrial offshore.



Pré-processamento

- Estados operacionais:
 - Estados originais: 6 (0 a 5);
 - Estados finais: 3 (normal, alerta, falha).
- Normalização robusta (RobustScaler) por sensor;
- Série temporal com amostras a cada 5 minutos.
- Segmentação em janelas:
 - `window_size = 10, window_step = 10`
 - janelas não sobrepostas (modo gerador local).

Definição do problema (modo gerador)

- Série multivariada: $x_t \in \mathbb{R}^d$, $t = 1, \dots, T$, com $d = 11$ sensores.
- Construímos janelas temporais:

$$\mathbf{X} = (x_{t-L+1}, \dots, x_t) \in \mathbb{R}^{L \times d}$$

com $L = 10$ passos de tempo.

- Objetivo: aprender um **modelo gerador** para as janelas \mathbf{X} :

$$p_\theta(\mathbf{X}) \approx p_{\text{dados}}(\mathbf{X})$$

- Modelo latente (VAE + difusão):
 - Encoder mapeia \mathbf{X} para um código latente z ;
 - Decoder reconstrói $\hat{\mathbf{X}}$ a partir de z ;
 - O espaço latente é regularizado para permitir **amostrar novos** z e gerar janelas sintéticas.
- Uso principal:
 - Geração de séries realistas para simulação de cenários e testes de algoritmos de monitoramento.

O modelo aprende a **recriar** e **amostrar** janelas de sensores compatíveis com o comportamento real do compressor.

Função de custo (modo gerador)

- O modelo admite uma função de custo multi-termo:

$$\mathcal{L} = \mathcal{L}_{\text{vae}} \quad \mathcal{L}_{\text{difusão}} = \lambda_m \mathcal{L}_{\text{miss}} + \lambda_v \mathcal{L}_{\text{vae}}$$

- O treinamento é **não supervisionado**, visando aprender:

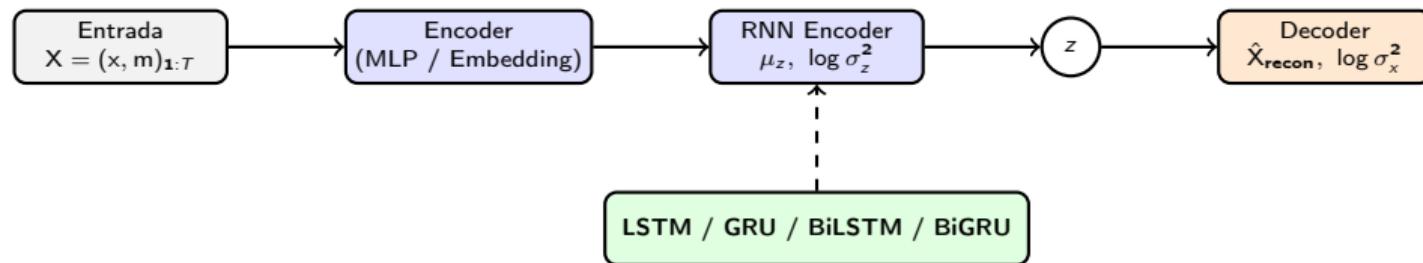
$$p_{\theta}(X) \approx p_{\text{dados}}(X)$$

$$\mathcal{L}_{\text{miss}} = \text{BCE}(m, \hat{m}) \quad \text{com} \quad \hat{m} = \sigma(f_{\text{miss}}(h))$$

$$\mathcal{L}_{\text{vae}} = \underbrace{\mathbb{E}_{q(z|X)}[-\log p(X | z)]}_{\text{erro de reconstrução}} + \beta \underbrace{\text{KL}(q(z | X) \| \mathcal{N}(0, I))}_{\text{regularização do espaço latente}}$$

- BCE entre a máscara verdadeira de observação e a máscara prevista, forçando o estado latente a codificar o padrão temporal de missing.
- O termo de reconstrução garante fidelidade aos sinais reais.
- O termo KL força um espaço latente contínuo e amostrável.
- Isso permite gerar novas janelas sintéticas de sensores com propriedades estatísticas similares às observadas.

Arquitetura base e arquiteturas avaliadas



- **Arquitetura base:** Encoder temporal recorrente acoplado a um VAE latente.
- A entrada X é codificada e passada ao **RNN encoder**, que parametriza $(\mu_z, \log \sigma_z^2)$.
- Amostramos z e o **decoder** reconstrói \hat{X} e estima a log-variância $\log \sigma_x^2$ para cada tempo/variável.
- **Arquiteturas avaliadas:**
 - **Deep MLP:** baseline feedforward (sem recorrência);
 - **LSTM:** LSTM unidirecional e BiLSTM;
 - **GRU:** GRU unidirecional e BiGRU.
 - **VAE:** Variational Autoencoder.

Variações de treinamento em relação à arquitetura base

Arquitetura base fixa: Encoder temporal BiLSTM + VAE latente (mesmo pipeline do slide anterior).

1. Variações de otimização

- **BiLSTM + Warmup & Scheduler:** Aumenta gradualmente a taxa de aprendizado no início e depois aplica um *scheduler* de decaimento, reduzindo instabilidades nas primeiras épocas e melhorando a convergência inicial.
- **BiLSTM + RAdam:** Substitui o Adam por **RAdam**, que corrige a variância adaptativa nos primeiros passos, tornando o treinamento mais estável.

2. Variações de regularização

- **BiLSTM + Variational Dropout:** Aplica *variational dropout* ($p = 0,2$) compartilhado no tempo, reduzindo overfitting sem quebrar dependências temporais.
- **BiLSTM/GRU + Difusão (missingness):** Acrescenta um objetivo probabilístico extra $\mathcal{L} = \lambda_m \mathcal{L}_{\text{miss}} + \lambda_v \mathcal{L}_{\text{vae}}$, forçando o modelo a ser robusto a *missing data* e a calibrar melhor incerteza.
- **BiLSTM + LayerNorm:** Aplica **LayerNorm** às ativações da BiLSTM, estabilizando as distribuições internas com efeito regularizante indireto.
- **BiLSTM + AdamW (L2/Weight Decay):** Usa **AdamW** com *weight decay* desacoplado ($\lambda = 10^{-4}$), acrescentando regularização L2 explícita nos pesos e melhorando generalização.

Configuração Experimental

- Divisão treino/teste: 80% / 20%.
- Dimensões do espaço latente foram validadas com a divisão 60% / 20% / 20%.
- Treinamento por 500 épocas, batch size 256.
- Paciência de 50 épocas para early stopping.
- Função de perda multi-tarefa conforme descrito.
- Otimização e Regularização tratadas como diferentes modelos.
- Otimizador: Adam (exceto variações).
- Taxa de aprendizado inicial: 3×10^{-4} .
- Early stopping baseado no NELBO do teste.
- Avaliação final no conjunto de teste.

- **NELBO** (Negative Evidence Lower Bound):

- Loss probabilística minimizada no treinamento;
- Maximiza implicitamente a verossimilhança (ELBO);
- Balanceia reconstrução e regularização latente (KL).
- $NELBO = \mathbb{E}_{q(z|x)}[-\log p(x|z)] + KL(q(z|x) \| p(z))$

- **NLL** (Negative Log-Likelihood):

- Generaliza o MSE ao modelar explicitamente a variância da distribuição predita;
- Penaliza erros grandes e variâncias mal calibradas (super ou subestimação de incerteza).

$$\bullet \text{NLL} = -\log p(x|\mu, \sigma^2) = \frac{(x-\mu)^2}{2\sigma^2} + \frac{1}{2} \log \sigma^2 + \text{cte}$$

Figuras de Mérito utilizadas

- **MSE** (Mean Squared Error):
 - Erro médio de reconstrução das séries;
 - Mede fidelidade gerativa.
- **Cobertura 90%** (cov_90):
 - Mede a fração de amostras reais que caem dentro do intervalo preditivo teórico [5%, 95%];
 - Avalia se o desvio padrão estimado produz uma **calibração probabilística consistente** com a cobertura nominal de 90%.
- **Largura do intervalo 90%** (width_90):
 - Corresponde à largura teórica do intervalo [5%, 95%] derivado da distribuição predita;
 - Quantifica a **sharpness** do modelo: intervalos menores indicam maior confiança, desde que a cobertura permaneça bem calibrada.

Resultados quantitativos

| Modelo | NELBO ↓ | NLL ↓ | MSE ↓ | Cov90 → 0.90 | Width90 ↓ |
|----------------------|--------------|--------------|----------------------|--------------|--------------|
| GRU Difusão | 0.510 | 0.456 | 0.237 ± 0.064 | 0.897 | 2.213 |
| BiLSTM Difusão | 0.518 | 0.462 | 0.243 ± 0.067 | 0.902 | 2.200 |
| GRU | 0.558 | 0.498 | 0.300 ± 0.108 | 0.905 | 2.252 |
| LSTM | 0.588 | 0.513 | 0.332 ± 0.129 | 0.900 | 2.251 |
| BiGRU | 0.589 | 0.499 | 0.335 ± 0.136 | 0.909 | 2.267 |
| BiLSTM VarDrop 0.2 | 0.595 | 0.506 | 0.317 ± 0.119 | 0.898 | 2.240 |
| BiLSTM RAdam | 0.597 | 0.511 | 0.341 ± 0.139 | 0.898 | 2.234 |
| BiLSTM Warmup/Sched. | 0.598 | 0.504 | 0.354 ± 0.146 | 0.903 | 2.251 |
| BiLSTM AdamW | 0.601 | 0.506 | 0.341 ± 0.141 | 0.900 | 2.224 |
| BiLSTM | 0.609 | 0.527 | 0.337 ± 0.133 | 0.901 | 2.269 |
| BiLSTM LayerNorm | 0.617 | 0.514 | 0.342 ± 0.140 | 0.907 | 2.281 |
| DEEP | 1.217 | 1.164 | 0.693 ± 0.209 | 0.916 | 2.785 |

Tabela: Comparação probabilística entre modelos. O critério principal de seleção é a minimização do NELBO. A métrica Cov90 avalia calibração probabilística, cujo valor ideal é próximo de 0.90. A métrica Width90 mede a precisão dos intervalos de previsão e é comparada apenas entre modelos com cobertura adequadamente calibrada.

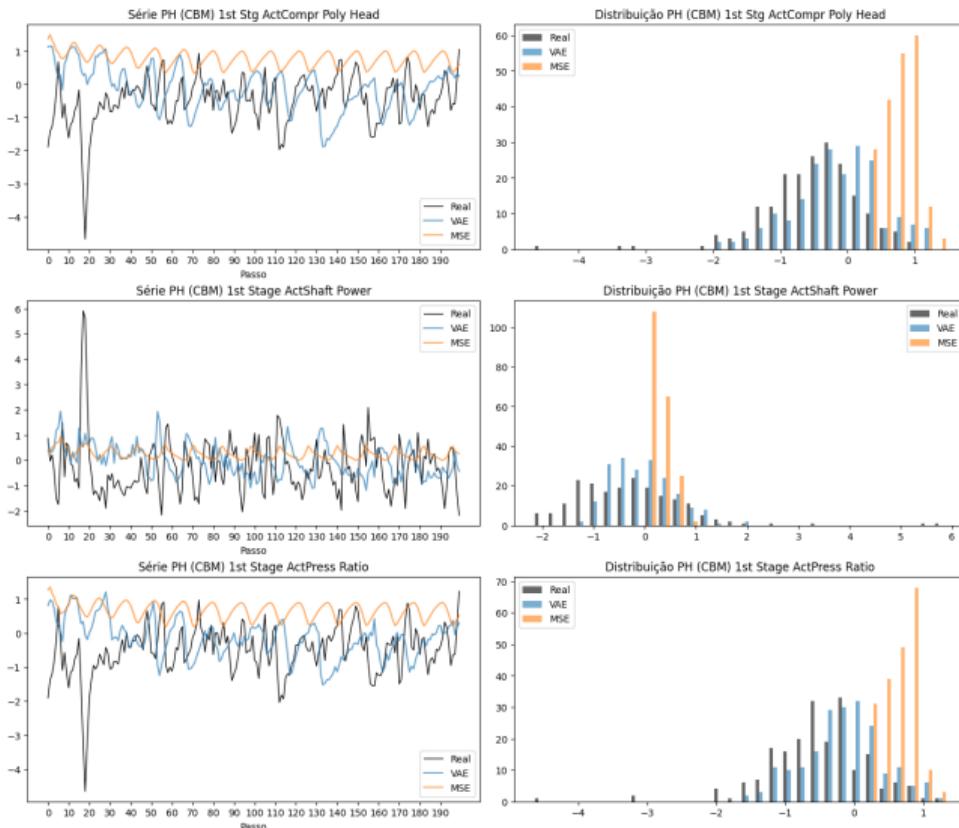
Curva de treinamento — Modelo campeão

Evolução do NELBO ao longo das épocas para o modelo **GRU Difusão**.

- Fase inicial instável, com forte variação do NELBO nas primeiras épocas;
- Convergência progressiva para um patamar de NELBO mais estável após dezenas de épocas;
- A melhor época (260) é definida por *early stopping*, privilegiando estabilidade e capacidade de generalização, e não o mínimo pontual da curva.



Geração de séries — Avaliação qualitativa



Comparação qualitativa

Séries temporais reais versus séries geradas pelo modelo variacional, comparadas a ajustes ponto-a-ponto baseados em MSE.

A abordagem probabilística captura melhor:

- a distribuição marginal dos sinais;
- a variabilidade temporal;
- a incerteza intrínseca dos dados.

- **Qualidade gerativa:**

- Modelos variacionais recorrentes superam o baseline feedforward;
- **GRU Difusão** apresenta o menor NELBO;

- **Calibração vs. sharpness:**

- Difusão gera intervalos mais estreitos (menor *width_90*);
- Cobertura próxima ao alvo de 90%;
- Trade-off equilibrado entre precisão e incerteza.

- **Impacto arquitetural:**

- A combinação **GRU + difusão** produz o melhor compromisso entre NELBO, NLL e MSE, com boa calibração (Cov90) e intervalos mais enxutos (Width90);

- **Treinamento e regularização:**

- Difusão teve melhor resultado como principal regularizador probabilístico;
- Dropout trouxe ganhos secundários;
- LayerNorm piorou o resultado, por destruir escalas absolutas necessárias ao VAE;
- Otimizadores afetam marginalmente o desempenho.

Conclusões (modo gerador)

- O uso de **RNNs com difusão** se mostrou superior como modelo gerador probabilístico para séries temporais industriais.
- Modelos recorrentes superam o baseline feedforward:
 - Fusão temporal melhora coerência de longo prazo.
- **Difusão é decisiva:**
 - **GRU com Difusão** é o modelo campeão, mas muito próximo do **BiLSTM com Difusão**;
 - Menor NELBO, NLL e MSE, com calibração adequada.
- O modelo permite:
 - Geração de séries realistas e calibradas;
 - Simulação de cenários raros ou críticos.
- Aplicações futuras:
 - Gêmeos digitais industriais;
 - Data augmentation orientada por incerteza.

Referências

- Kingma, D. P.; Welling, M. *Auto-Encoding Variational Bayes*. ICLR, 2014.
- Hochreiter, S.; Schmidhuber, J. *Long Short-Term Memory*. Neural Computation, 1997.
- Cho et al. *Learning Phrase Representations using RNN Encoder–Decoder*. EMNLP, 2014.
- Ho, J.; Jain, A.; Abbeel, P. *Denoising Diffusion Probabilistic Models*. NeurIPS, 2020.
- Che et al. *Recurrent Neural Networks for Multivariate Time Series with Missing Values*. Sci Rep, 2018.