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

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 generativos 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.

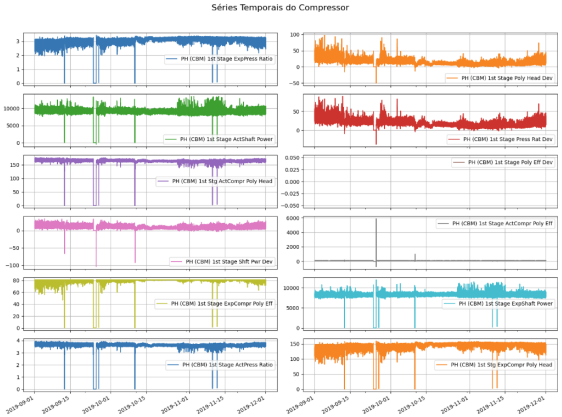
Comparar arquiteturas:

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

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

  com $L = 10$ passos de tempo.
- Objetivo: aprender um **modelo gerador** para as janelas X:

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

- Modelo latente (VAE + difusão):
  - Encoder mapeia X para um código latente z;
  - Decoder reconstrói $\hat{X}$ 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:

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

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

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

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

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

- 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.

Arquiteturas avaliadas

- **TSDF_DEEP**:
  - Modelo feedforward sobre janelas (baseline não-recorrente).
- **TSDF_LSTM**:
  - RNN LSTM unidirecional;
  - Versão **BiLSTM**.
- **TSDF_GRU**:
  - RNN GRU unidirecional;
  - Versão **BiGRU**.

Arquitetura baseline: TSDF_DEEP (feedforward)

- Entrada: janela temporal $X = (x_1, \ldots, x_L)$, com $x_t \in \mathbb{R}^{11}$ e máscara $m_t \in \{0,1\}^{11}$.
- Codificação inicial por MLP (por timestep):

$$e_t = \phi\left(W_e[x_t \,\|\, m_t] + b_e\right), \quad t = 1, \ldots, L$$

  onde $\phi(\cdot)$ é uma ativação não linear (ReLU/GELU).
- Incorporação explícita do tempo:

$$e'_t = \mathrm{MLP}\left([e_t \,\|\, t_t]\right)$$

  com t_t o timestamp normalizado do instante t.
- Representação latente da janela:
  - Obtida independentemente por timestep (sem estados recorrentes);
  - Integra informação temporal apenas via timestamps explícitos.
- Cabeças de saída (modo gerador):

$$\hat{x}_t = f_\theta(e'_t) \quad \text{(reconstrução / geração)}$$

- Treinamento:
  - Otimizador Adam;
  - Mesmas funções de perda usadas nos modelos recorrentes.

Arquitetura 1: LSTM

- Entrada: janela temporal $\mathrm{X} = (x_1, \ldots, x_L)$, $x_t \in \mathbb{R}^{11}$.
- Codificação temporal com LSTM unidirecional:

$$(h_t, c_t) = \mathrm{LSTM}(x_t, h_{t-1}, c_{t-1}), \quad t = 1, \ldots, L$$

- Representação latente da janela:

$$z = h_L \in \mathbb{R}^H$$

- z representa um código latente contínuo da dinâmica temporal.
- Decoder:

$$\hat{\mathrm{X}} = p_\theta(\mathrm{X} \mid z)$$

- Otimização:
  - Otimizador Adam, minimizando perda conjunta (classificação de estado + regressão de tempo).

Arquitetura 2: GRU

- Mesma entrada:
$$\mathrm{X} = (x_1, \ldots, x_L), \quad x_t \in \mathbb{R}^{11}$$
- Codificação temporal recorrente:
$$\{h_t\}_{t=1}^{L} = \mathrm{GRU}(\mathrm{X})$$
- Representação latente da janela:
$$z = h_L \in \mathbb{R}^H$$
- z representa um resumo contínuo da dinâmica temporal da janela.
- Reconstrução:
$$\hat{\mathrm{X}} = p_\theta(\mathrm{X} \mid z)$$
- Treinamento:
  - Otimização por Adam, com os mesmos hiperparâmetros do baseline Deep.

Arquitetura 3: BiGRU + GRU fuser

- Codificação da janela com BiGRU:

$$h_t^{\rightarrow}, h_t^{\leftarrow} = \mathrm{BiGRU}(x_t, h_{t-1}^{\rightarrow}, h_{t+1}^{\leftarrow})$$

$$z_k = [h_L^{\rightarrow} \parallel h_1^{\leftarrow}] \in \mathbb{R}^{2H}$$

  onde z_k é o embedding da k-ésima janela.

- Fusão temporal entre janelas com GRU fuser:

$$m_k = \mathrm{GRU}_{\mathsf{fuser}}(z_k, m_{k-1}), \quad k = 1, \ldots, K$$

- Fuser captura dependências de longo alcance entre janelas, produzindo um latente m_K adequado à geração.
- Cabeças de saída a partir de m_K:

$$\hat{\mathrm{X}} \sim p_\theta(\mathrm{X} \mid m_K)$$

- Otimização:
  - Adam sobre todos os parâmetros (BiGRU + GRU fuser + cabeças).

Arquitetura 4: BiLSTM + GRU fuser

- Codificação da janela com BiLSTM:

$$(h_t^{\rightarrow}, c_t^{\rightarrow}), (h_t^{\leftarrow}, c_t^{\leftarrow}) = \mathrm{BiLSTM}(x_t, \dots)$$

$$z_k = [h_L^{\rightarrow} \parallel h_1^{\leftarrow}] \in \mathbb{R}^{2H}$$

- Fusão temporal entre janelas com GRU fuser:

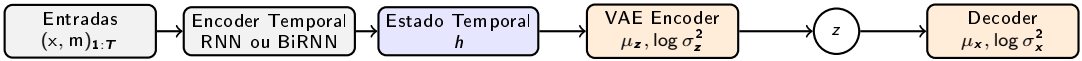
$$m_k = \mathrm{GRU}_{\mathsf{fuser}}(z_k, m_{k-1}), \quad k = 1, \dots, K$$

- Fuser captura dependências de longo alcance entre janelas, produzindo um latente m_K adequado à geração.
- Cabeças de saída:

$$\hat{\mathrm{X}} \sim p_\theta(\mathrm{X} \mid m_K)$$

- Otimização:
  - Treino conjunto com Adam, mesma função de perda multi-tarefa.

Pipeline principal: Encoder Temporal + VAE Latente



- Arquitetura base é composta de um encoder temporal uni ou bidirecional seguido de um espaço latente variacional, responsável por capturar incerteza e permitir geração probabilística das séries.

Variações de treinamento - Otimização

- **BiLSTM + Warmup & Scheduler**
  - Aumenta gradualmente a taxa de aprendizado no início do treino, evitando instabilidades e melhorando a convergência inicial.
- **BiLSTM + RAdam**
  - Reduz a variância adaptativa do Adam nos primeiros passos, tornando o treinamento mais estável.

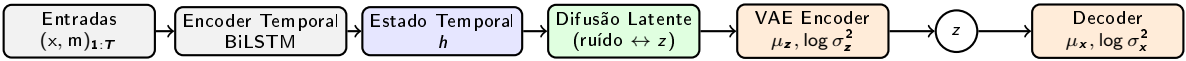
Variações de treinamento — Regularização

- **BiLSTM + Variational Dropout**
  - Regularização explícita no tempo, reduzindo overfitting sem quebrar dependências temporais.
  - Variational_dropout utilizado: 0.2
- **BiLSTM + Difusão (missingness)**
  - Regularização implícita via objetivos probabilísticos, forçando robustez a dados ausentes.
  - $\mathcal{L} = \lambda_m \mathcal{L}_{\text{miss}} + \lambda_v \mathcal{L}_{\text{vae}}$
- **BiLSTM + LayerNorm**
  - Estabiliza as ativações internas, com efeito regularizante indireto.
- **BiLSTM + AdamW (L2/Weight Decay)**
  - Regularização explícita dos pesos, melhorando generalização.
  - L2/weight_decay utilizado: 1e-4

Configuração Experimental

- Divisão treino/teste: 80% / 20%.
- A quantidade de neurônios já foi validada anteriormente em outra disciplina
- Treinamento por 500 épocas, batch size 256.
- Paciênica 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.

Arquitetura vencedora — BiLSTM + Difusão Latente (Geração)



- No modo gerador, a difusão modela a incerteza no espaço latente, aumentando robustez, diversidade e estabilidade estatística das séries reconstruídas.

Figuras de Mérito utilizadas

- **NELBO** (Negative Evidence Lower Bound):
  - Loss probabilística minimizada no treinamento;
  - Maximiza implicitamente a verossimilhança (ELBO);
  - Balanceia reconstrução e regularização latente (KL).
  - $NELBO = \mathbb{E}_{q(z|x)}[-\log p(x \mid z)] + \mathrm{KL}(q(z \mid x) \,\|\, p(z))$
- **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).
  - $\mathrm{NLL} = -\log p(x \mid \mu, \sigma^2) \quad = \frac{(x-\mu)^2}{2\sigma^2} + \frac{1}{2}\log \sigma^2 + \mathrm{cte}$

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

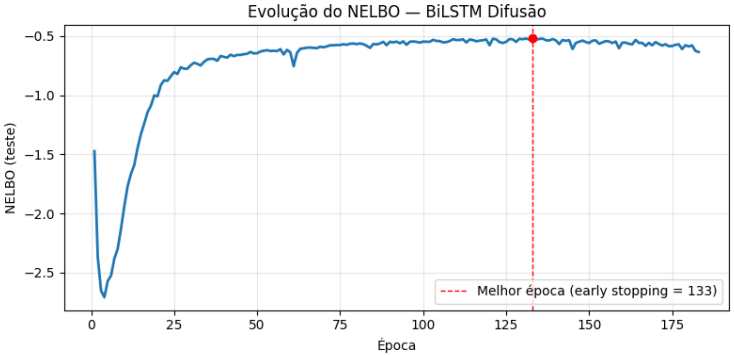
Modelo	NELBO ↓	NLL ↓	MSE ↓	Cov90 → 0.90	Width90 ↓
BiLSTM Difusão	**0.518**	**0.462**	**0.243 ± 0.067**	**0.902**	**2.200**
GRU	0.558	0.498	0.300 ± 0.108	0.905	2.252
LSTM	0.588	0.513	0.332 ± 0.129	0.900	2.251
BiGRU	0.589	0.499	0.335 ± 0.136	0.909	2.267
BiLSTM VarDrop 0.2	0.595	0.506	0.317 ± 0.119	0.898	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.

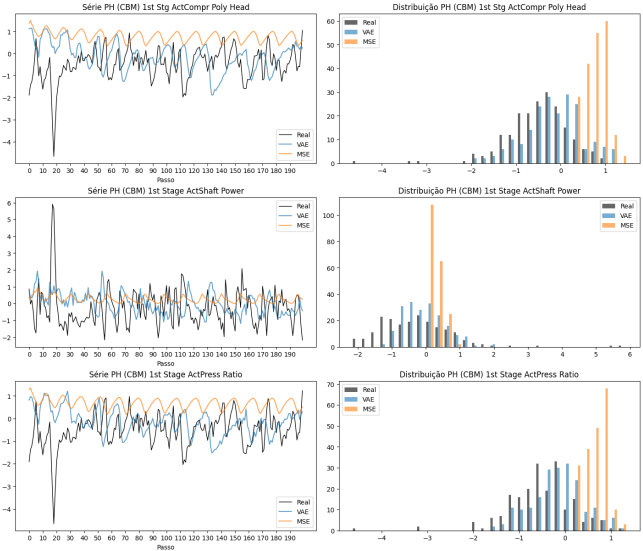
Curva de treinamento — Modelo campeão

Evolução do NELBO ao longo das épocas para o modelo **BiLSTM 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 (133) 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.

Discussão dos resultados (modo gerador)

- **Qualidade gerativa**:
  - Modelos variacionais recorrentes superam o baseline feedforward;
  - **BiLSTM Difusão** apresenta o menor NELBO;
  - Redução consistente de NLL e MSE frente a LSTM/GRU unidirecionais.
- **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**:
  - Bidirecionalidade melhora as representações latentes;
  - *GRU fuser* aumenta coerência entre janelas;
  - Maior profundidade temporal favorece geração estável.
- **Treinamento e regularização**:
  - Difusão atua como principal regularizador probabilístico;
  - Dropout e LayerNorm trazem ganhos secundários;
  - Otimizadores afetam marginalmente o desempenho.

Conclusões (modo gerador)

- O framework **TSDF** é eficaz como modelo gerador probabilístico para séries temporais industriais.
- Modelos recorrentes superam o baseline feedforward:
  - Arquiteturas bidirecionais aprendem latentes mais ricos;
  - Fusão temporal melhora coerência de longo prazo.
- **Difusão é decisiva**:
  - **BiLSTM Difusão** é o modelo campeão;
  - 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.