

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

Rodrigo Petrus Domingues

CPE727 – Deep Learning

6 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 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

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

Séries Temporais do Compressor



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:

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

com $L = 10$ passos de tempo.

- Objetivo: aprender um **modelo gerador** para as janelas 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 **recrutar e amostrar** janelas de sensores compatíveis com o comportamento real do compressor.

Função de custo (modo gerador)

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

$$\mathcal{L} = \lambda_1 \mathcal{L}_{\text{rec}} + \lambda_2 \mathcal{L}_{\text{diff}} + \lambda_3 \mathcal{L}_{\text{time}} + \lambda_4 \mathcal{L}_{\text{mask}} + \boxed{\lambda_5 \mathcal{L}_{\text{VAE-x}}} + \lambda_6 \mathcal{L}_{\text{VAE-}\tau}$$

- Neste experimento, o foco é exclusivamente gerador:

$$\lambda_5 = 1, \quad \lambda_{1,2,3,4,6} = 0$$

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

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

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

- 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.

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

Hiperparâmetros principais

- `in_channels = 11`
- `hidden_dim = 11 * 32`
- `status_dim = 3`
- Janela: `window_size = 10, window_step = 10`

Arquitetura baseline: TSDF_DEEP (feedforward)

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

$$e_t = \phi(W_e[x_t \parallel m_t] + b_e), \quad t = 1, \dots, L$$

onde $\phi(\cdot)$ é uma ativação não linear (ReLU/GELU).

- Incorporação explícita do tempo:

$$e'_t = \text{MLP}([e_t \parallel t_t])$$

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

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

- Representação latente da janela:

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

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

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

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

Arquitetura 2: GRU

- Mesma entrada: $X = (x_1, \dots, x_L)$, $x_t \in \mathbb{R}^{11}$.
- Codificação temporal com GRU unidirecional:

$$h_t = \text{GRU}(x_t, h_{t-1}), \quad t = 1, \dots, L$$

- Representação latente da janela:

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

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

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

- Otimização:
 - Otimizador Adam, com os mesmos hiperparâmetros do baseline Deep (learning rate, batch, etc.).

Arquitetura 3: BiGRU + GRU fuser

- Codificação da janela com BiGRU:

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

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

onde z_k é o embedding da k -ésima janela.

- Fusão temporal entre janelas com GRU fuser:

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

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

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

- Otimização:

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

Arquitetura 4: BiLSTM + GRU fuser

- Codificação da janela com BiLSTM:

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

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

- Fusão temporal entre janelas com GRU fuser:

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

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

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

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

- **BiLSTM + Warmup & Scheduler**

- Aumenta gradualmente a taxa de aprendizado no início do treino, evitando instabilidades e melhorando a convergência inicial.
- `warmup_steps = 50, min_lr_factor = 0.01`

- **BiLSTM + RAdam**

- Reduz a variância adaptativa do Adam nos primeiros passos, tornando o treinamento mais estável.
- `optimizer_name = 'radam'`

• BiLSTM + Variational Dropout

- Regularização explícita no tempo, reduzindo overfitting sem quebrar dependências temporais.
- `variational_dropout = 0.2`

• BiLSTM + Difusão (missingness)

- Regularização implícita via objetivos probabilísticos, forçando robustez a dados ausentes.
- `lam = [0., 0.0, 0.0, 0.05, 0.0, 0.95]`, `rebuild=True`

• BiLSTM + LayerNorm

- Estabiliza as ativações internas, com efeito regularizante indireto.
- `use_layernorm = True`

• BiLSTM + AdamW (Weight Decay)

- Regularização explícita dos pesos, melhorando generalização.
- `optimizer_name = 'adamw'`, `weight_decay = 1e-4`

Configuração Experimental

- Divisão treino/validação/teste: 60% / 20% / 20%.
- Treinamento por 500 épocas, batch size 256.
- Paciência de 50 épocas para early stopping.
- Função de perda multi-tarefa conforme descrito.
- Otimizador: Adam (exceto variações).
- Taxa de aprendizado inicial: 1×10^{-3} .
- Early stopping baseado em MSE de validação.
- Avaliação final no conjunto de teste.

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).
- **MSE** (Mean Squared Error):
 - Erro médio de reconstrução das séries;
 - Mede fidelidade gerativa.
- **NLL** (Negative Log-Likelihood):
 - Qualidade probabilística da geração;
 - Penaliza variância mal calibrada.
- **Cobertura 90%** (cov_90):
 - Mede se os intervalos gerados contêm os valores reais;
 - Avalia calibração do gerador.
- **Largura do intervalo 90%** (width_90):
 - Avalia sharpness do modelo;
 - Intervalos estreitos com boa cobertura são desejáveis.

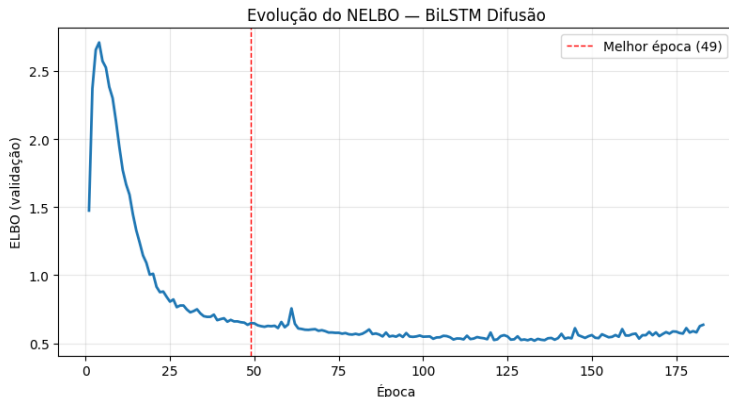
Resultados quantitativos

Modelo	NELBO ↓	MSE ↓	NLL ↓	Cov90 → 0.90	Width90 ↓
BiLSTM Difusão	0.518	0.243 ± 0.067	0.462	0.902	2.200
GRU	0.558	0.300 ± 0.108	0.498	0.905	2.252
LSTM	0.588	0.332 ± 0.129	0.513	0.900	2.251
BiGRU	0.589	0.335 ± 0.136	0.499	0.909	2.267
BiLSTM VarDrop 0.2	0.595	0.317 ± 0.119	0.506	0.898	2.240
BiLSTM RAdam	0.597	0.341 ± 0.139	0.511	0.898	2.234
BiLSTM Warmup/Sched.	0.598	0.354 ± 0.146	0.504	0.903	2.251
BiLSTM AdamW	0.601	0.341 ± 0.141	0.506	0.900	2.224
BiLSTM	0.609	0.337 ± 0.133	0.527	0.901	2.269
BiLSTM LayerNorm	0.617	0.342 ± 0.140	0.514	0.907	2.281
DEEP	1.217	0.693 ± 0.209	1.164	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**.
- Observa-se:
 - Convergência mais rápida;
 - Menor oscilação durante o treinamento;
 - Melhor estabilidade em comparação aos modelos sem normalização.



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.