

Utilização de RNNs para Predição de mudança de estado em compressor industrial

Fernanda Mickosz Villa Verde

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).
- Mudanças de regime operacional (estados) afetam: Desempenho energético; Risco de falha; Planejamento de manutenção.
- Predizer **quando** ocorrerá a próxima mudança de estado é útil para:
 - Detecção precoce de desvios;
 - Replanejamento operacional;
 - Suporte à decisão para operadores.

Objetivos do trabalho:

- Modelar séries temporais industriais (compressores) com amostragem irregular, dados faltantes e alta dimensionalidade utilizando arquiteturas recorrentes.
- Predição dos **estados** em janelas futuras;
- Predição do **tempo até a próxima mudança de estado**.

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, log-likelihood e layernorm).

Base de dados e pré-processamento

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;
- Amostragem a cada 5 minutos;
- Janelas temporais:
 - $L = 40$ amostras ($\approx 3h20min$);
 - Deslocamento de 40 amostras (janelas não sobrepostas).

Definição do problema

- Série multivariada: $\mathbf{x}_t \in \mathbb{R}^d$, $t = 1, \dots, T$, com $d = 11$ sensores.
- Estados discretos: $s_t \in \{0, 1, 4\}$ (após fusão dos 6 estados iniciais).
- Para cada janela $X = (\mathbf{x}_{t-L+1}, \dots, \mathbf{x}_t)$, queremos:
 - Predizer a distribuição sobre estados futuros:

$$p(s_{t+\Delta} | X)$$

- Predizer o tempo até a próxima mudança de estado:

$$\hat{\tau} = f_{\theta}(X)$$

- Problema multi-tarefa:
 - Classificação de estados (estado futuro) + regressão de tempo (tempo até a mudança).

A partir de janelas de histórico de sensores, o modelo aprende simultaneamente a prever o próximo regime operacional e o tempo restante até a próxima mudança de estado.

Função de custo

- Função de custo multi-tarefa:

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

- \mathcal{L}_{rec} : erro de reconstrução da série observada (ajuste aos sinais reais).
- $\mathcal{L}_{\text{diff}}$: perda de difusão, regularizando a predição de ruído (robustez a incerteza).
- $\mathcal{L}_{\text{time}}$: erro na predição do tempo até a próxima mudança de estado.
- $\mathcal{L}_{\text{mask}}$: aprendizado explícito do padrão de dados ausentes.
- $\mathcal{L}_{\text{VAE-x}}$: regularização variacional da reconstrução dos sinais.
- $\boxed{\mathcal{L}_{\text{VAE-}\tau}}$: **modelagem probabilística do tempo até falha**, penalizando mudanças próximas e calibrando incerteza.

$$\mathcal{L}_{\text{VAE-}\tau} = \mathbb{E}_{q(z_\tau | \mathbf{x})} \left[-\log p(\tau | z_\tau) \right] + \beta \text{KL}(q(z_\tau) \| \mathcal{N}(0, I))$$

- Peso maior atribuído a instantes próximos à mudança de estado:

$$\log p(\tau | z_\tau) \propto w_t \cdot \|\hat{\tau} - \tau\|^2, \quad w_t \gg 1$$

- Permite estimar **incerteza, intervalos de confiança e cobertura probabilística**.

- **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 = 40, window_step = 40`

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 (multi-tarefa):

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

$$\hat{\tau} = g_\theta(e'_t) \quad (\text{tempo até mudança de estado})$$

- 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 da janela:

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

- Cabeças de saída (multi-tarefa):

$$\hat{y}_{\text{state}} = \text{softmax}(W_s z + b_s)$$

$$\hat{\tau} = W_\tau z + b_\tau$$

- 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 da janela:

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

- Cabeças de saída:

$$\hat{y}_{\text{state}} = \text{softmax}(W_s z + b_s)$$

$$\hat{\tau} = W_\tau z + b_\tau$$

- 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:

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

$m_K \in \mathbb{R}^{H_f}$ resume a história recente de janelas.

- Cabeças de saída a partir de m_K :

$$\hat{y}_{\text{state}} = \text{softmax}(W_s m_K + b_s)$$

$$\hat{\tau} = W_{\tau} m_K + b_{\tau}$$

- 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:

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

- Cabeças de saída:

$$\hat{y}_{\text{state}} = \text{softmax}(W_s m_K + b_s)$$

$$\hat{\tau} = W_{\tau} m_K + b_{\tau}$$

- 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):
 - Mede o erro médio quadrático na predição do tempo até a próxima mudança;
 - Avalia precisão pontual das estimativas.
- **NLL** (Negative Log-Likelihood):
 - Avalia a qualidade probabilística das previsões;
 - Penaliza incertezas mal calibradas (sub ou superestimação).
- **Cobertura 90%** (cov_90):
 - Fração de observações reais contidas no intervalo preditivo de 90%;
 - Mede a calibração do modelo.
- **Largura do intervalo 90%** (width_90):
 - Largura média dos intervalos preditivos de 90%;
 - Reflete o compromisso entre precisão e incerteza (sharpness).

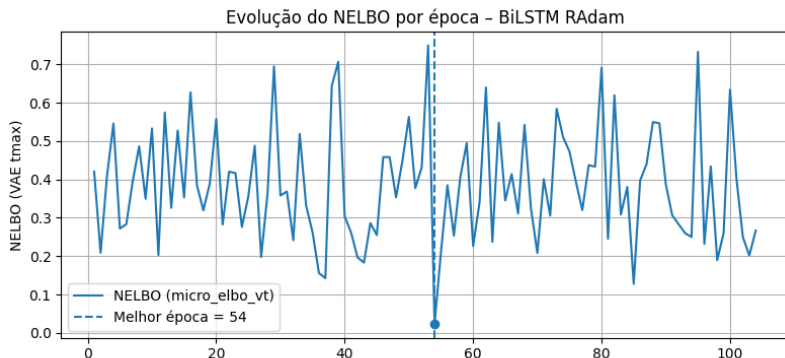
Resultados quantitativos

Modelo	NELBO ↓	MSE (micro) ↓	NLL ↓	Cov90 → 0.90	Width90 ↓
BiLSTM RAdam	0.023	0.002 ± 0.000	0.023	0.946	0.270
BiLSTM Warmup/Sched.	0.033	0.002 ± 0.000	0.033	0.946	0.270
LSTM	0.064	0.002 ± 0.000	0.063	0.937	0.270
BiLSTM	0.150	0.002 ± 0.000	0.150	0.920	0.270
BiGRU	0.203	0.001 ± 0.000	0.202	0.875	0.270
BiLSTM AdamW	0.230	0.001 ± 0.001	0.229	0.866	0.270
BiLSTM Difusão	0.329	0.002 ± 0.001	0.328	0.839	0.270
BiLSTM LayerNorm	0.383	0.054 ± 0.001	0.332	0.839	0.270
GRU	0.464	0.003 ± 0.001	0.463	0.812	0.270
BiLSTM VarDrop 0.2	0.992	0.002 ± 0.001	0.991	0.705	0.270
DEEP	8016.760	0.819 ± 0.011	33.711	0.250	0.270

Tabela: Comparação de desempenho entre modelos e variações.

Curva de treinamento — Modelo BiLSTM RAdam

- Curva de NELBO bastante oscilatória, refletindo o desbalanceamento entre grupos e a raridade de mudanças de estado;
- Após as primeiras dezenas de épocas, o modelo passa a operar em um patamar de NELBO baixo, com mínimos recorrentes;
- A melhor época (**epoch 54**) atinge o menor NELBO entre as épocas avaliadas, sendo o melhor compromisso entre ajuste e complexidade para a tarefa de mudança de estado.



- **Qualidade probabilística:**

- O **BiLSTM RAdam** apresenta o menor NELBO, indicando melhor ajuste probabilístico global;
- Bons valores de NLL e MSE micro confirmam alta precisão na predição de mudanças de estado raras.

- **Calibração das incertezas:**

- Cobertura próxima de 0.90 nos melhores modelos indica boa calibração probabilística;
- Width90 constante sugere intervalos estáveis e comparáveis entre arquiteturas.

- **Impacto arquitetural:**

- Modelos bidirecionais superam LSTM/GRU unidirecionais;
- Em cenários fortemente desbalanceados, o principal desafio é a **otimização sob gradientes raros**, e não o overfitting;
- Otimizadores adaptativos (RAdam, warmup) estabilizam o treinamento e facilitam a captura de eventos de mudança raros, enquanto regularização excessiva tende a diluir esses sinais.

- A predição de mudanças de estado é um problema fortemente desbalanceado e sensível à calibração probabilística.
- Arquiteturas recorrentes probabilísticas são essenciais:
 - Capturam dependências temporais longas;
 - Modelam explicitamente incertezas associadas ao evento de transição.
- O **BiLSTM RAdam** é o modelo mais adequado:
 - Menor NELBO entre os modelos avaliados;
 - Boa calibração ($\text{Cov90} \approx 0.95$) e baixo erro micro.
- Esses resultados indicam potencial para:
 - Detecção antecipada de mudanças de regime;
 - Suporte a sistemas de monitoramento e alerta industrial.