RNNs em Séries Temporais: Arquiteturas, Treinamento e Aplicações

Fernanda Villa Verde, Leonardo Britto, Luiza Helena e Rodrigo Petrus

Modelagem de Séries Temporais com RNNs - LSTM / GRU / BiLSTM / BiGRU

13 de novembro de 2025

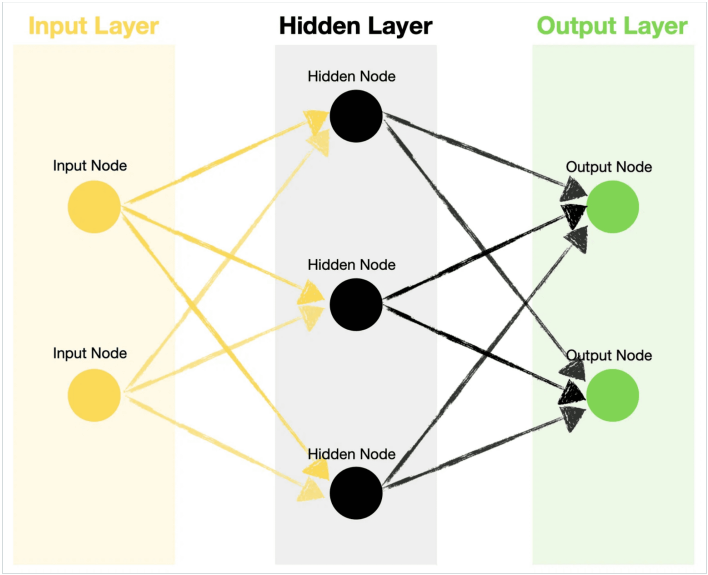
Agenda

Motivação e Contexto Histórico

Por que RNNs? As redes feedforward assumem entradas independentes e processam cada amostra isoladamente. Entretanto, muitas tarefas reais exigem **modelagem temporal**:

- Séries financeiras e sensoriais
- Áudio e fala
- Texto e linguagem natural
- Sequências de vídeo

Avanço conceitual: As **Redes Neurais Recorrentes (RNNs)** introduziram a noção de **memória de estados anteriores**, permitindo representar dependências temporais ao longo da sequência.



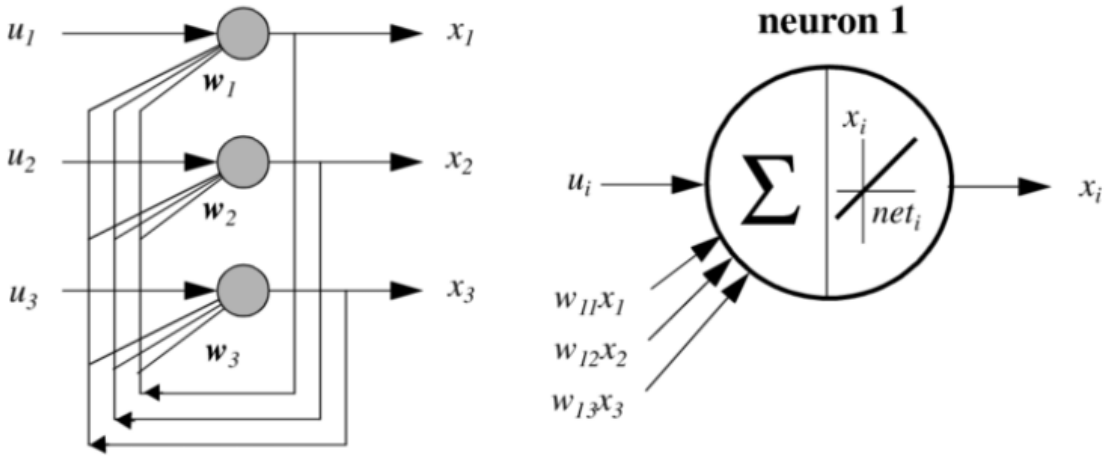
RNN: fluxo temporal de estados com memória.

Linha do Tempo das RNNs

Ano	Autor(es)	Contribuição
1982	Hopfield	Rede recorrente como memória associativa (não sequencial).
1986	Jordan	Recorrência via **saída** anterior.
1990	Elman	Recorrência via **estado oculto** (RNN padrão).
1990	Werbos	Formaliza **BPTT** para treinar RNNs.
1991	Hochreiter	Demonstra o **vanishing gradient**.
1997	Hochreiter & Schmidhuber	Introduzem **LSTM** (estado de célula preserva gradiente).
1997	Schuster & Paliwal	Introduzem **BiRNN** (contexto passado + futuro).
2005	Graves & Schmidhuber	Combinação: **LSTM + BiRNN**.
2014	Cho et al.	Introduzem **GRU** (simplifica LSTM).
2014	Chung, J.	Combinação: **GRU + BiRNN**.

Hopfield Network (1982) - Memória associativa [1]

Ideia central: A Hopfield Network é uma rede recorrente totalmente conectada e simétrica, capaz de armazenar e recuperar padrões — funcionando como uma **memória associativa**.



Rede totalmente conectada: cada neurônio recebe sinais dos demais.

Resumo conceitual:

- **Conexões simétricas:** garantem estabilidade e convergência.
- **Atualização recorrente:** cada neurônio depende do estado dos outros.
- **Mínimo de energia:** estado fixo da rede corresponde a uma memória recuperada.
- **Padrões aprendidos:** configurações estáveis codificadas nos pesos.

A Hopfield Network introduz o princípio de memória associativa — base conceitual das RNNs.

Jordan Network (1986) [2]

Ideia central: memória via **feedback da saída** em *context units*.

Contexto (copia saída anterior):

$$c_t = y_{t-1}$$

Estado oculto:

$$h_t = \phi(W_{xh}\, x_t + W_{ch}\, c_t + b_h)$$

Saída:

$$y_t = \psi(W_{hy}\, h_t + b_y)$$

Onde: ϕ tipicamente tanh (ou ReLU) e ψ pode ser softmax (classificação) ou identidade (regressão).

Resumo: a **memória** vem de y_{t-1} (não de h_{t-1}).

Elman (1990) - RNN: Simple Recurrent Network (SRN) [3]

Ideia central: recorrência no **estado oculto** — a RNN "padrão".

Estado oculto (recorrente):

$$h_t = \phi(W_{xh}\, x_t + W_{hh}\, h_{t-1} + b_h)$$

Saída:

$$y_t = \psi(W_{hy}\, h_t + b_y)$$

Notação:

- x_t: entrada no tempo t
- h_t: estado oculto (memória)
- y_t: saída predita
- W_{hh}: pesos de transição (estado $\rightarrow$ estado)
- W_{xh}: pesos de entrada (entrada $\rightarrow$ estado)
- W_{hy}: pesos de saída (estado $\rightarrow$ saída)
- b_h, b_y: vieses

Inicialização típica: $h_0 = 0$ (ou parâmetro treinável).
Observações:

- ϕ costuma ser tanh; ψ pode ser softmax (classificação).
- Base para variantes modernas (LSTM/GRU) que mitigam *vanishing/exploding gradients*.

Werbos (1990) — Backpropagation Through Time (BPTT) [4]

Forward:

$$h_t = \phi(W_{xh}x_t + W_{hh}h_{t-1} + b_h), \qquad y_t = \psi(W_{hy}h_t + b_y)$$

Perda total:

$$L = \sum_{t=1}^{T} \ell(y_t, \tilde{y}_t)$$

BPTT — o gradiente retropropaga no tempo:

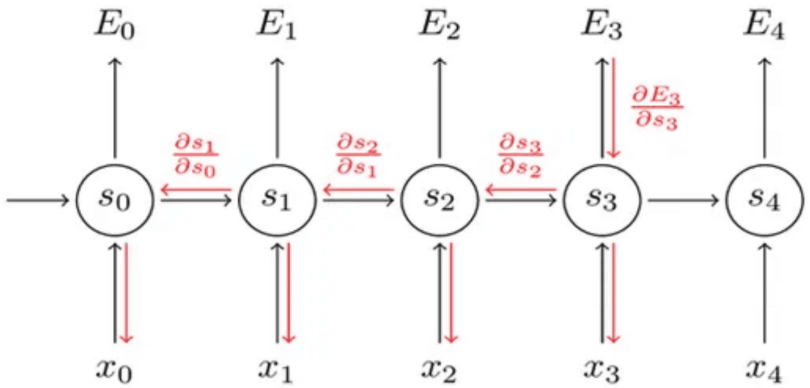
$$\frac{\partial L}{\partial h_t} = \frac{\partial L}{\partial y_t}\frac{\partial y_t}{\partial h_t} + \frac{\partial L}{\partial h_{t+1}}\frac{\partial h_{t+1}}{\partial h_t}$$

$$\frac{\partial h_{t+1}}{\partial h_t} = D_{t+1}W_{hh}, \quad D_{t+1} = \mathrm{diag}(1 - \tanh^2(\cdot))$$

$$\frac{\partial L}{\partial W_{hh}} = \sum_{t=1}^{T} \left(\frac{\partial L}{\partial h_t}\right)(h_{t-1})^\top$$

Ponto-chave: O erro "viaja para trás" através do tempo, acumulando dependências.

Werbos (1990) — Backpropagation Through Time (BPTT) [4] (cont.)



A RNN vista ao longo do tempo. Cada estado s_t influencia o próximo.

Ideia central

- A RNN processa a sequência passo a passo.
- Cada estado oculto h_t guarda uma **memória** do passado.
- O passado influencia o futuro através da recorrência.

Por que isso é importante?

- Permite modelar dependências temporais.
- Útil para texto, fala, séries temporais, sensores, controle, etc.

Resumo mental:

estado novo = f(entrada atual, lembrança do passado)

Hochreiter (1991) — Vanishing Gradient [5]

Equação-chave (forma simplificada do gradiente):

$$\frac{\partial L}{\partial h_t} = \frac{\partial L}{\partial h_T} \prod_{k=t+1}^{T} \frac{\partial h_k}{\partial h_{k-1}}$$

Para uma RNN simples com tanh:

$$\frac{\partial h_k}{\partial h_{k-1}} = D_k \, W_{hh}, \quad \text{onde} \quad D_k = \operatorname{diag}(1 - \tanh^2(\cdot)), \quad \|D_k\|_2 \leq 1.$$

Consequência:

$$\left\| \frac{\partial h_k}{\partial h_{k-1}} \right\|_2 \leq \|W_{hh}\|_2 \quad \Rightarrow \quad \left\| \frac{\partial L}{\partial h_t} \right\| \lesssim \|W_{hh}\|_2^{T-t}$$

Interpretação:

- Se $\|W_{hh}\|_2 < 1$: o gradiente **decai exponencialmente** $\rightarrow$ **vanishing gradient**.
- Se $\|W_{hh}\|_2 > 1$: o gradiente pode **explodir**.
- Esse efeito **não é de implementação**, mas da **matemática da recorrência**.

Hochreiter & Schmidhuber (1997) — LSTM [6]

Ideia central: introduzir um **estado de célula** c_t com **rota de gradiente quase constante**, permitindo **memória de longo prazo** e evitando **vanishing gradient**.

Equações da LSTM:

$$
\begin{aligned}
i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i) && \text{(porta de entrada)}\\
f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f) && \text{(porta de esquecimento)}\\
\tilde{c}_t &= \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) && \text{(conteúdo candidato)}\\
c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c}_t && \text{(estado de célula)}\\
o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o) && \text{(porta de saída)}\\
h_t &= o_t \odot \tanh(c_t) && \text{(estado oculto)}
\end{aligned}
$$

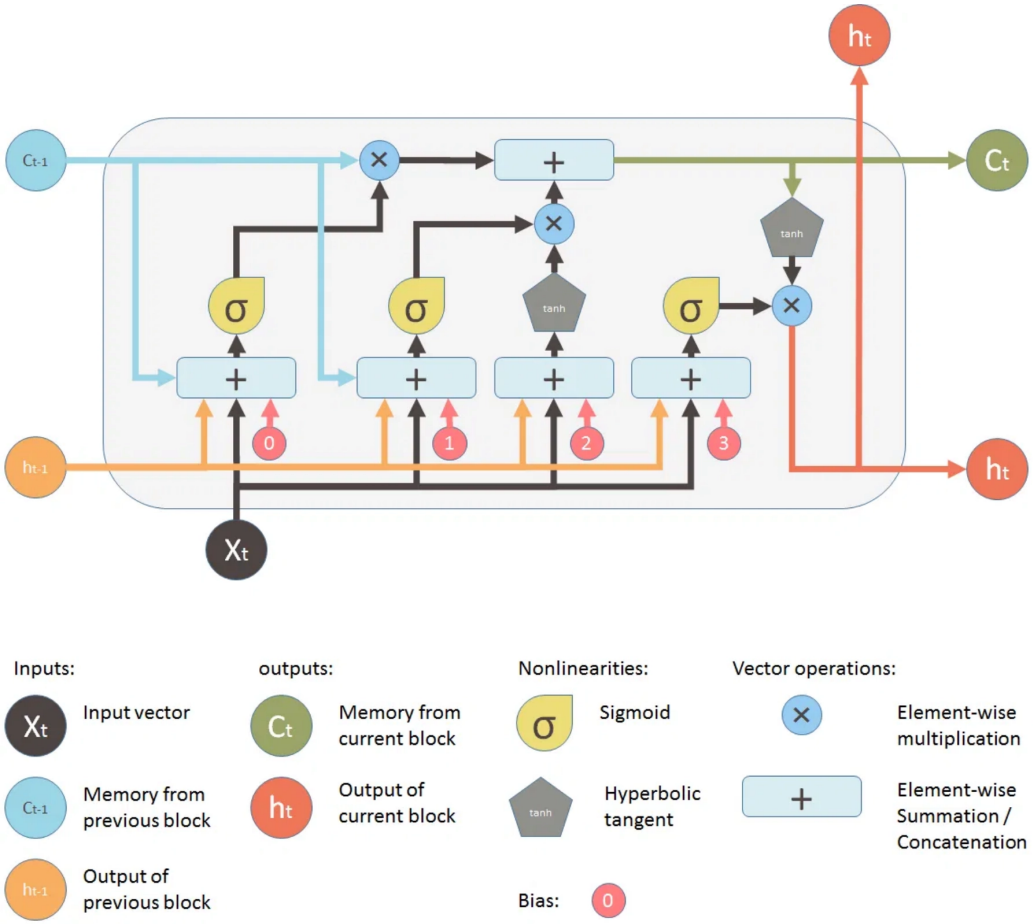
Por que isso evita vanishing gradient?

$$\frac{\partial c_t}{\partial c_{t-1}} = f_t$$

Se $f_t \approx 1$, então:

$$\frac{\partial L}{\partial c_t} \approx \frac{\partial L}{\partial c_{t-1}} \quad \Rightarrow \quad \text{gradiente flui sem decair}$$

Hochreiter & Schmidhuber (1997) — LSTM (cont.) [6]



Resumo:

- A LSTM adiciona **portas** para controlar escrita, esquecimento e leitura da memória.
- O estado de célula c_t funciona como **rota de gradiente preservada**.
- Resolve o problema identificado por **Hochreiter (1991)**.

Schuster & Paliwal (1997) — RNN Bidirecional (BiRNN) [7]

Ideia central: processar a sequência **nos dois sentidos** — um fluxo temporal **forward** ($t = 1 \to T$) e outro **backward** ($t = T \to 1$), permitindo que cada posição tenha acesso ao **passado e ao futuro**.

Equações da BiRNN:

$$\overrightarrow{h}_t = \phi\left(W_x^{\rightarrow} x_t + W_h^{\rightarrow} \overrightarrow{h}_{t-1} + b^{\rightarrow}\right)$$

$$\overleftarrow{h}_t = \phi\left(W_x^{\leftarrow} x_t + W_h^{\leftarrow} \overleftarrow{h}_{t+1} + b^{\leftarrow}\right)$$

Fusão (saída no tempo t):

$$h_t = [\,\overrightarrow{h}_t\,;\,\overleftarrow{h}_t\,] \quad \text{(concatenação, mais comum)}$$

Interpretação:

- A rede **forward** captura contexto do **passado**.
- A rede **backward** captura contexto do **futuro**.
- A combinação fornece **representações mais ricas por tempo** t.

Aplicações típicas:

- Rotulação de sequência (POS tagging, NER, chunking)
- Reconhecimento de fala e alinhamento temporal (CTC, seq2seq)
- Modelos de atenção e embeddings contextuais (pré-transformer)

Graves & Schmidhuber (2005) - BiLSTM — Bidirectional LSTM [8]

Ideia central: aplicar uma **LSTM** no sentido **forward** e outra no sentido **backward**, de modo que cada passo temporal tenha acesso a **passado + futuro**.

LSTM forward:

$$\overrightarrow{h}_t, \overrightarrow{c}_t = \text{LSTM}_{\rightarrow}(x_t, \overrightarrow{h}_{t-1}, \overrightarrow{c}_{t-1})$$

LSTM backward:

$$\overleftarrow{h}_t, \overleftarrow{c}_t = \text{LSTM}_{\leftarrow}(x_t, \overleftarrow{h}_{t+1}, \overleftarrow{c}_{t+1})$$

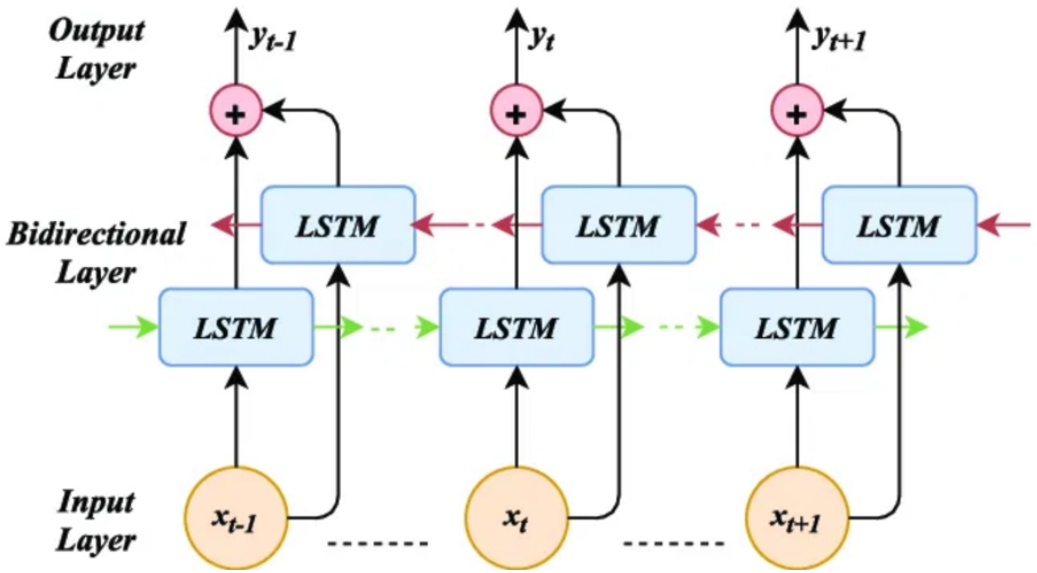
Fusão (saída no tempo t):

$$h_t = [\overrightarrow{h}_t ; \overleftarrow{h}_t] \quad \text{(concatenação, mais comum)}$$

Interpretação:

- $\overrightarrow{h}_t$: informações do **passado**.
- $\overleftarrow{h}_t$: informações do **futuro**.
- A fusão fornece uma representação **contextual bidirecional**.

BiLSTM — Long Short-Term Memory Bidirecional (cont.) [8]



BiLSTM

Por que utilizar BiLSTM?

- Mantém a **memória de longo prazo** da LSTM.
- Excelente para tarefas onde o **contexto futuro é relevante**.
- Muito utilizada em **NLP** (POS tagging, NER, parsing), **fala**, **bio-sinais**.

Cho et al. (2014) — GRU (Gated Recurrent Unit) [9]

Ideia central: simplificar a LSTM mantendo a capacidade de **memória de longo prazo**, reduzindo o número de portas e **sem estado de célula separado** (h_t = memória).

Equações da GRU:

$$z_t = \sigma(W_{xz}x_t + W_{hz}h_{t-1} + b_z) \quad \text{(porta de atualização)}$$

$$r_t = \sigma(W_{xr}x_t + W_{hr}h_{t-1} + b_r) \quad \text{(porta de reset)}$$

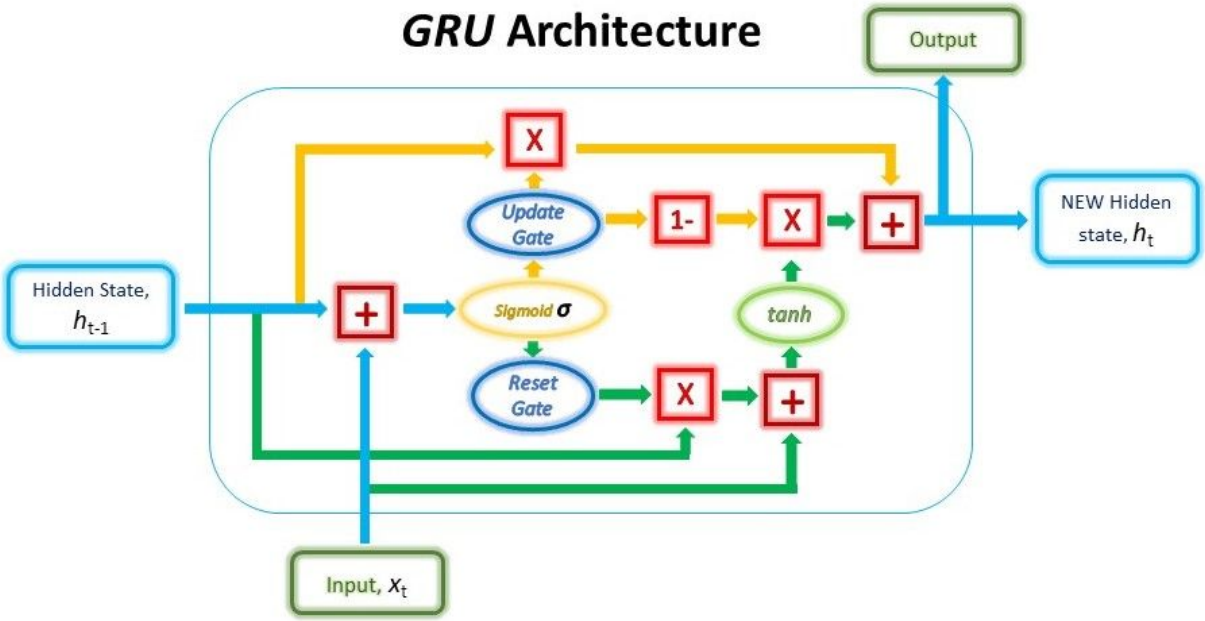
$$\tilde{h}_t = \tanh(W_{xh}x_t + W_{hh}(r_t \odot h_{t-1}) + b_h) \quad \text{(conteúdo candidato)}$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \quad \text{(estado oculto atualizado)}$$

Interpretação:

- z_t decide **quanto do passado manter**.
- r_t decide **quanto ignorar do passado** ao propor novo conteúdo.
- Não possui estado de célula c_t: **memória é mantida em** h_t.

Cho et al. (2014) — GRU (Gated Recurrent Unit) (cont.) [9]



Por que GRU ficou popular?

- Menos parâmetros que LSTM → **mais leve**.
- Desempenho comparável em muitas tarefas sequenciais.
- Treino mais estável que RNN simples (não sofre vanishing severo).

Chung, J. (2014) - BiGRU — Gated Recurrent Unit Bidirecional [10]

Ideia central: aplicar uma **GRU** no sentido **forward** e outra no sentido **backward**, combinando seus estados ocultos para obter **contexto passado + futuro**.

GRU forward:

$$\overrightarrow{h}_t = \mathrm{GRU}_{\rightarrow}(x_t, \overrightarrow{h}_{t-1})$$

GRU backward:

$$\overleftarrow{h}_t = \mathrm{GRU}_{\leftarrow}(x_t, \overleftarrow{h}_{t+1})$$

Fusão (saída no tempo t):

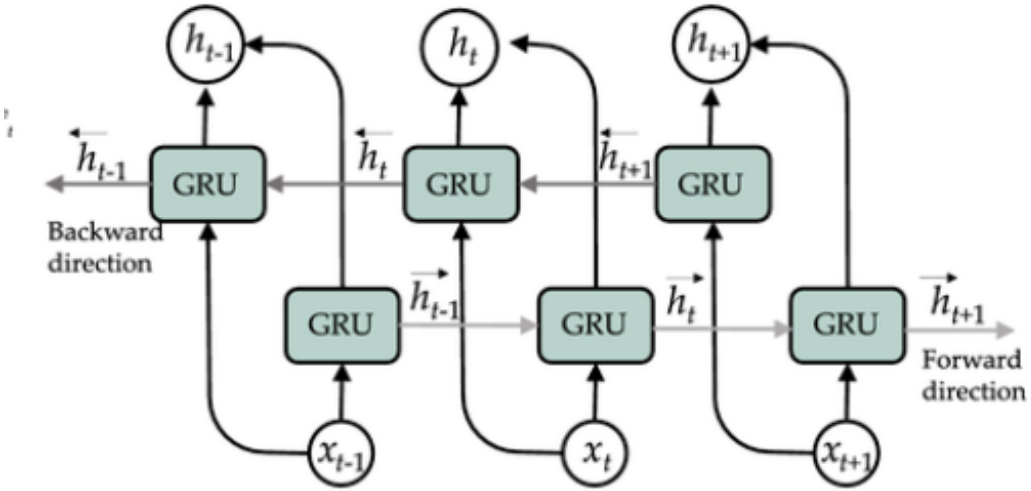
$$h_t = [\overrightarrow{h}_t ; \overleftarrow{h}_t] \quad \text{(concatenação)}$$

Interpretação:

- $\overrightarrow{h}_t$ captura dependências **anteriores** na sequência.
- $\overleftarrow{h}_t$ captura dependências **futuras**.
- A concatenação fornece uma **representação mais rica** em cada passo temporal.

Chung, J. (2014) - BiGRU — Gated Recurrent Unit Bidirecional (cont.)[10]

BiGRU



Por que utilizar BiGRU?

- Mantém as vantagens da GRU (menos parâmetros que LSTM).
- Boa estabilidade em sequências longas.
- Muito utilizada em **NLP, fala, anotação de sequência, NER, segmentação**.

Arquiteturas Bidirecionais: Acoplada vs. Desacoplada ("Zig–Zag") [11]

Objetivo: Capturar dependências temporais **passado** $\leftrightarrow$ **futuro**.

1) BiRNN Acoplada (Convencional)

- **Forward** e **Backward** recebem **a mesma entrada** x_t.
- São processadas **em paralelo** e **independentes** entre si.
- A fusão ocorre **após** o cálculo dos dois estados:

$$\overrightarrow{h}_t = f(x_t, \overrightarrow{h}_{t-1}), \quad \overleftarrow{h}_t = f(x_t, \overleftarrow{h}_{t+1}), \quad z_t = [\overrightarrow{h}_t; \overleftarrow{h}_t].$$

- **Usada apenas offline**, pois depende de informação futura.

2) BiRNN Desacoplada (Zig–Zag / Teacher–Student)

- O ramo **backward é treinado primeiro** (usa contexto futuro).
- Depois, a rede **forward é treinada usando** x_t **e** a representação do backward como **sinal professor**:

$$\overleftarrow{h}_t = f(x_t, \overleftarrow{h}_{t+1}) \quad \text{(treino offline)}$$
$$\overrightarrow{h}_t = g(x_t, \overrightarrow{h}_{t-1}, \overleftarrow{h}_t) \quad \text{(treino do aluno)}$$

- Na **inferência**, somente $\overrightarrow{h}_t$ é usado $\rightarrow$ **modelo causal**.

Acoplada = duas direções independentes em paralelo.
Desacoplada = backward como ***professor*** $\rightarrow$ *forward* ***aprende a prever sem ver o futuro.***

Técnicas de Fusão Bidirecional pós-BiRNN:

- Concatenação Direta
- Fusão gated
- Fusão recorrente (Fuser RNN)

Fusão Bidirecional pós-BiRNN: Concatenação direta [7, 8]

Após processar a sequência com uma BiRNN (BiLSTM/BiGRU), precisamos combinar as representações das duas direções em um vetor único para cada passo de tempo t

$$z_t = [\, h_t^{\rightarrow}; h_t^{\leftarrow} \,]$$

- Simples, padrão em BiLSTM/BiGRU.
- Mantém toda a informação das duas direções.
- Limitação: trata passado e futuro como igualmente relevantes, sem controle dinâmico.

Fusão Bidirecional pós-BiRNN: Fusão *gated* (seleção dinâmica) [11]

$$g_t = \sigma\left(W_g[\,h_t^{\rightarrow}; h_t^{\leftarrow}\,] + b_g\right)$$

$$z_t = g_t \odot h_t^{\rightarrow} \;+\; (1 - g_t) \odot h_t^{\leftarrow}$$

g_t : vetor de pesos entre 0 e 1 (gate dinâmico)

W_g, b_g : parâmetros aprendidos

$\odot$: produto elemento a elemento

- A rede aprende quando confiar mais no histórico causal ($h_t^{\rightarrow}$) ou no contexto futuro ($h_t^{\leftarrow}$).
- Melhora robustez em sinais ruidosos.

Fusão Bidirecional pós-BiRNN: Fusão recorrente (*fuser RNN*) [11]

$$z_t = [\, h_t^{\rightarrow}; h_t^{\leftarrow} \,]$$

$$u_t = \mathrm{GRU}_{\mathrm{fuser}}(z_t, u_{t-1})$$

- Uma GRU (ou LSTM) adicional unidirecional processa a sequência já fundida.
- Atua como um "refinador temporal": suaviza, agrega contexto longo e gera uma representação causal u_t pronta para uso em produção on-line.
- Permite treinar com contexto bidirecional, mas implantar só o ramo forward distilado.

Comparativo de Estratégias de Fusão Bidirecional

Método	Mecânica de Fusão	Vantagens Principais	Aplicações Típicas
Concatenação	Combina diretamente os estados das direções: $z_t = [h_t^{\rightarrow}; h_t^{\leftarrow}]$	• Simples e eficiente. • Mantém toda a informação temporal. • Facilmente integrável a MLPs ou CNNs.	• Classificação e regressão offline. • Representação de contexto completo. • Modelos base ou ablação.
Fusão Gated	Calcula um gate dinâmico: $z_t = g_t \odot h_t^{\rightarrow} + (1 - g_t) \odot h_t^{\leftarrow}$ onde $g_t = \sigma(W_g [h_t^{\rightarrow}; h_t^{\leftarrow}])$	• Adapta a contribuição de cada direção por timestep. • Robusto a ruído e desbalanceamento temporal. • Oferece interpretabilidade (importância causal/anticausal).	• Séries ruidosas (sensores, texto clínico). • Diagnóstico temporal e detecção de eventos.
Fuser RNN	Empilha uma RNN extra sobre a fusão: $u_t = \mathrm{GRU}_{\mathbf{fuser}}([h_t^{\rightarrow}; h_t^{\leftarrow}], u_{t-1})$	• Agrega contexto de longo prazo. • Suaviza e refina a sequência. • Permite distilação causal para deploy online.	• Modelos industriais/tempo real. • Reconstrução e previsão contínua. • "Teacher–student" bidirecional → causal.

Concat = simples, Gate = adaptativo, Fuser = refinamento temporal e causal.

Regularização e Estabilização em RNNs

O treinamento de **Redes Neurais Recorrentes (RNNs)** exige técnicas específicas para lidar com **instabilidades de gradiente** e **overfitting**, mantendo a coerência temporal e a convergência estável do modelo.

Objetivos principais:

- Evitar explosão/vanishing de gradientes.
- Melhorar generalização em sequências longas.
- Estabilizar ativações internas e acelerar o aprendizado.

Técnicas essenciais:

1. **Gradient Clipping [12]** — limita a norma do gradiente ($\|g\|_2 \leq c$) para evitar atualizações instáveis.
2. **Variational Dropout [13]** — aplica a mesma máscara de dropout ao longo do tempo, preservando a coerência temporal.
3. **Layer Normalization [14]** — normaliza as ativações por camada e timestep, reduzindo flutuações internas e melhorando a estabilidade numérica.

Essas estratégias combinadas formam a base para o treinamento robusto de RNNs, LSTMs e GRUs em contextos de sequências longas e ruído nos dados.

Pascanu, Mikolov & Bengio (2013) - Gradient Clipping [12]

Motivação: Em redes recorrentes (RNNs) e arquiteturas profundas, o problema do **exploding gradient** faz com que os gradientes cresçam descontroladamente, levando a instabilidade numérica e divergência no treinamento.

Ideia principal: Limitar a norma do gradiente — *gradient norm clipping* — mantendo o vetor de gradientes dentro de um raio c.

$$g = \nabla_\theta L \qquad \text{se } \|g\|_2 > c, \quad g \leftarrow \frac{c}{\|g\|_2} g$$

Intuição:

- Previne saltos muito grandes na superfície de perda.
- Garante que a atualização permaneça num "raio seguro" em torno dos parâmetros.
- Contribui para estabilidade numérica, e facilita o treinamento de redes profundas ou recorrentes.

Aplicações comuns:

- RNNs / LSTMs para sequências longas.
- Treinamento de redes profundas com muitas camadas.
- Cenários onde gradientes muito grandes causam divergência.

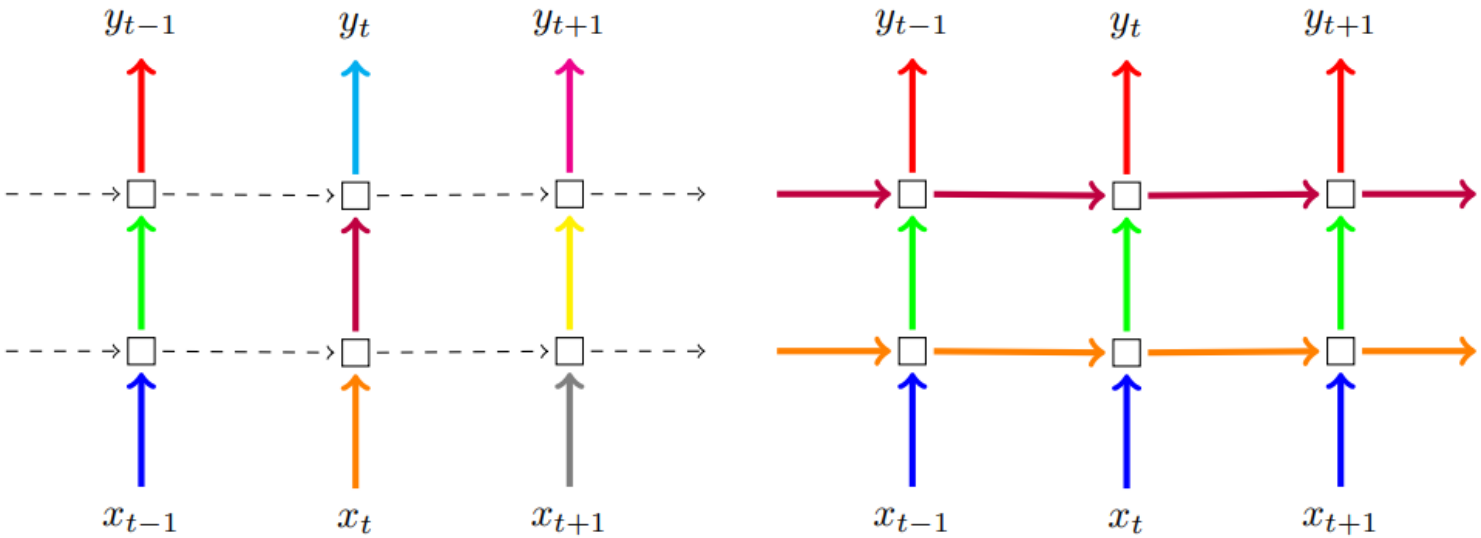
Gal & Ghahramani (2016) - Variational Dropout [13]

Problema do dropout tradicional em RNNs:
Se uma **máscara diferente** for aplicada em cada passo temporal:

- o ruído muda a cada timestep,
- **desestabiliza a memória** da rede.

Solução: Variational Dropout

- A **mesma máscara de dropout** é mantida para toda a sequência.
- Preserva a coerência temporal.
- Reduz overfitting sem destruir o estado recorrente.



(a) Naive dropout RNN (b) Variational RNN

Gal & Ghahramani (2016) - Variational Dropout (cont.) [13]

Em frameworks modernos:

$$h_t = f\left(W_{hh}\left(d_h \odot h_{t-1}\right) + W_{xh}\, x_t\right)$$

h_t	: vetor de estado oculto no tempo t
x_t	: entrada no tempo t
W_{hh}, W_{xh}	: matrizes de pesos recorrentes e de entrada
d_h	: máscara de dropout, $d_h \sim \text{Bernoulli}(p)$
p	: probabilidade de retenção (ex: $p = 0.8$)
$\odot$	: produto elemento a elemento

Ba, Kiros & Hinton (2016) - Layer Normalization (LN) [14]

- Normaliza as ativações dentro de cada camada, em cada passo de tempo, de forma independente do tamanho do batch:

$$\mathrm{LN}(h_t) = \gamma \frac{h_t - \mu_t}{\sqrt{\sigma_t^2 + \varepsilon}} + \beta \quad \text{com } \varepsilon > 0 \text{ pequeno.}$$

h_t	: vetor de ativações da camada no tempo t
μ_t, σ_t	: média e desvio padrão das ativações em h_t
γ, β	: parâmetros aprendidos de escala e deslocamento

- Reduz instabilidade do gradiente e acelera a convergência, especialmente em sequências longas.
- Em LSTMs/GRUs, é aplicada nas transformações lineares de cada porta:

$$i_t = \sigma(\mathrm{LN}(W_i[h_{t-1}, x_t]) + b_i)$$

i_t	: porta de entrada (input gate)
W_i	: pesos da porta de entrada
$[h_{t-1}, x_t]$	: concatenação do estado anterior e da entrada atual
b_i	: termo de viés (bias)

O mesmo processo é aplicado às demais portas (f, o, g), mantendo escalas consistentes entre todas as transformações internas.

Evolução para Arquiteturas Modernas

A evolução das arquiteturas de processamento de sequências representa uma das trajetórias mais importantes no aprendizado profundo. Partindo das RNNs clássicas, o campo testemunhou inovações fundamentais que culminaram nos modelos de linguagem modernos.

- **2014 - Sequence-to-Sequence (Seq2seq) [15]:** Sutskever et al. introduzem arquitetura encoder-decoder com RNNs/LSTMs para tradução automática neural, revolucionando o campo
- **2015 - Mecanismo de Atenção [16]:** Bahdanau et al. propõem atenção para resolver limitação do vetor de contexto fixo, permitindo que o decoder "foque" em partes relevantes da entrada.
- **2017 - Transformer [17]:** Vaswani et al. introduzem arquitetura baseada puramente em atenção (self-attention), eliminando recorrência e permitindo paralelização massiva.
- **2018 - BERT e GPT [18, 19]:** Modelos de linguagem pré-treinados baseados em Transformer (BERT para encoding bidirecional, GPT para geração autoregressiva) alcançam state-of-the-art em NLP.
- **2020 - Large Language Models (LLMs) [20, 21]:** GPT-3, GPT-4, LLaMA, PaLM e outros LLMs com bilhões de parâmetros dominam tarefas de linguagem natural através de scaling e prompting.

Comparação entre LSTM, GRU, BiLSTM e BiGRU

RNNs permanecem relevantes em nichos específicos: dispositivos embarcados com recursos limitados, processamento em tempo real com baixa latência, e aplicações onde paralelização não é crítica.

Característica	LSTM	GRU	BiLSTM	BiGRU
Portas	3 (forget, input, output)	2 (reset, update)	3 (duas direções)	2 (duas direções)
Estados internos	C_t, h_t	h_t	C_t, $\overrightarrow{h}_t$, $\overleftarrow{h}_t$	$\overrightarrow{h}_t$, $\overleftarrow{h}_t$
Contexto capturado	Passado	Passado	Passado e futuro	Passado e futuro
Nº de parâmetros	Alto	$\sim$25% menor	$\sim 2\times$ o da LSTM	$\sim 2\times$ o da GRU
Velocidade de treino	Mais lenta	Mais rápida	Mais lenta (duas passagens)	Razoável (duas passagens)
Capacidade de modelagem	Alta	Boa	Muito alta	Alta
Aplicações típicas	Séries longas	Modelos leves	NLP, voz, texto bidir.	NLP leve, embeddings

Resumo: Modelos bidirecionais (BiLSTM/BiGRU) exploram dependências temporais em ambas as direções. GRU e BiGRU são mais leves; LSTM e BiLSTM tendem a capturar relações mais complexas. Arquiteturas híbridas combinando RNNs e atenção também emergem como soluções eficientes

Conclusões Principais

1) Contribuição Conceitual das RNNs

- Introduzem a ideia de **estado oculto** como **memória dinâmica**.
- Permitem modelar séries temporais, fala, texto e qualquer dado sequencial.
- Fundamentam a passagem de redes estáticas $\rightarrow$ modelos temporais.

2) Contribuição Matemática

- BPTT torna explícita a **dependência temporal no gradiente**.
- Vanishing/Exploding não é falha de implementação — é da **recorrência**.
- LSTM e GRU criam **caminhos estáveis** para o gradiente ao longo do tempo.

3) Arquiteturas Modernas

- BiLSTM/BiGRU incorporam **contexto futuro** $\rightarrow$ melhores embeddings.
- Ainda são eficazes quando há **causalidade**, **baixa latência** ou **poucos dados**.
- **Transformers não substituem RNNs**: estendem a ideia com **atenção global**.

Mensagem Final: RNNs são o **ponto de transição** entre redes neurais clássicas e modelos de atenção. Entender suas equações é entender **o alicerce** do processamento moderno de sequências.

Cenários de Uso em Séries Temporais

Problema: Modelar séries temporais industriais (compressores) com amostragem irregular, dados faltantes e alta dimensionalidade. Dois cenários: **predição** e **reconstrução** utilizando arquiteturas recorrentes (RNNs, LSTM, GRU, BiRNNs).

Estratégias adotadas:

- Avaliação de modelos clássicos e bidirecionais.
- Teste de três técnicas de fusão pós-BiRNN (concat/ gated / fuser).
- Uso de regularização temporal: dropout variacional, layer normalization e gradient clipping.

Base de dados: Cognite — Sinais reais de sensores de um compressor industrial offshore.

Loss: Predição e Reconstrução de Séries Temporais

- **Predição** Loss: MSE no último timestamp de cada janela:

$$\mathcal{L} = \frac{1}{\sum_{t=1}^{T} m_t} \sum_{t=1}^{T} m_t \cdot (x_t - \hat{x}_t)^2$$

- **Reconstrução** Loss: Último timestamp da janela + difusão de ausência L1 — NLL Gaussiano ponderado por $\hat{\lambda}_t$:

$$\text{sse}_t = \sum_c \left(\hat{x}_{t,c} - x_{t,c}\right)^2, \qquad \hat{\lambda}_t = \text{softplus}\left(W_\lambda h_t + b_\lambda\right)$$

$$\log p(x_t \mid h_t) = -\tfrac{1}{2}\,\hat{\lambda}_t\,\text{sse}_t + \tfrac{1}{2}\,n\,\log\hat{\lambda}_t - \tfrac{1}{2}\,n\log(2\pi)$$

$$\mathcal{L}_1 = -\sum_t \log p(x_t \mid h_t)$$

Metodologia de Treinamento e Avaliação

Figura de Mérito:

- Micro MSE (Mean Squared Error)

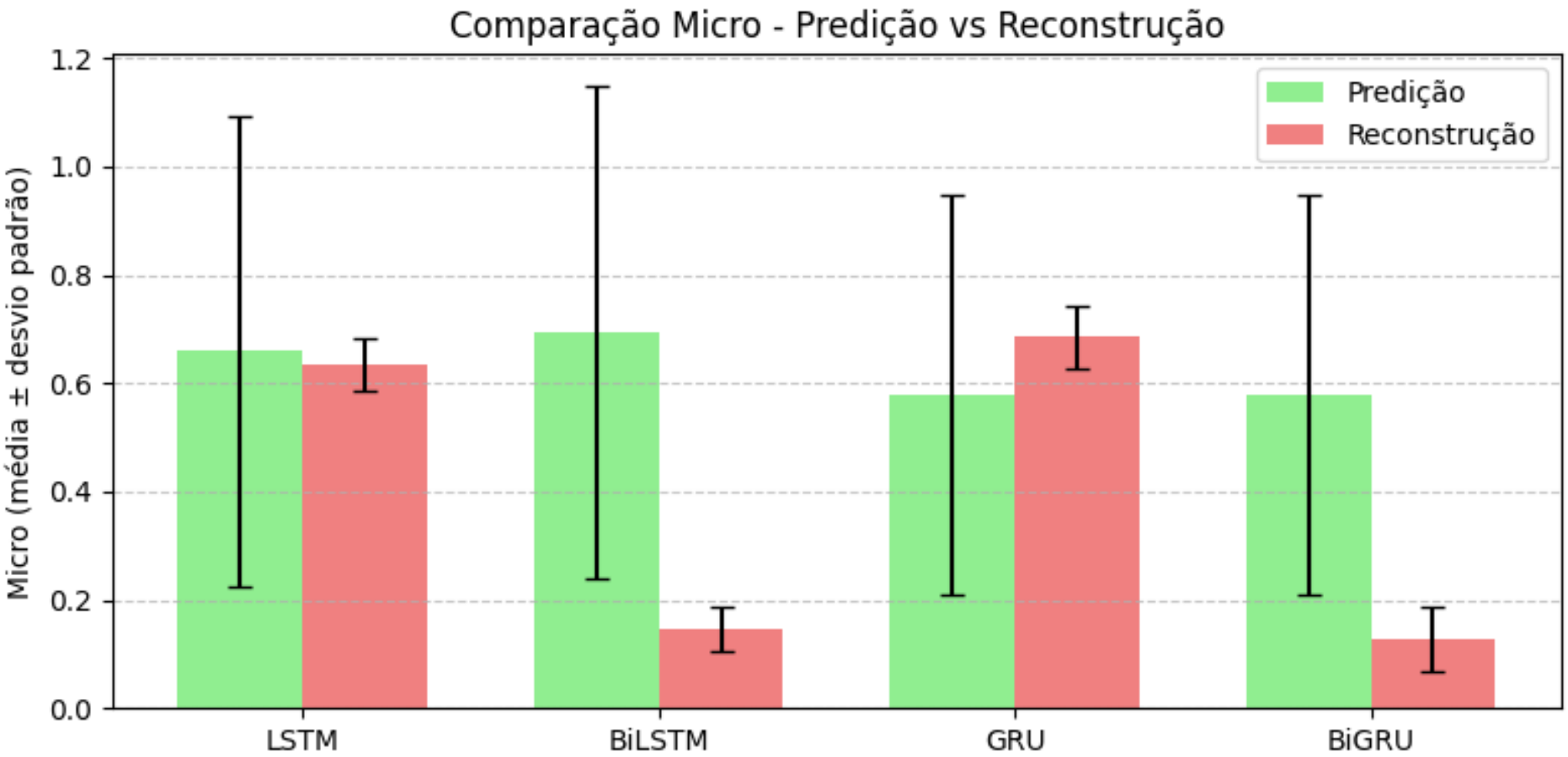
Setup de Treino:

- Divisão de dados: 60% treino, 20% validação, 20% teste.
- Otimizador: Adam W
- Warm up
- Schedular
- Weight decay (L2 regularization).
- Early stopping com paciência de $\sim$20 épocas para predição e 50 épocas para reconstrução.

Metodologia de Treinamento e Avaliação

PREDIÇÃO: lOSS MSE RECONSTRUÇÃO: Loss INSERÇÃO DE DIFUSÃO/AUSÊNICA LAYER NORM

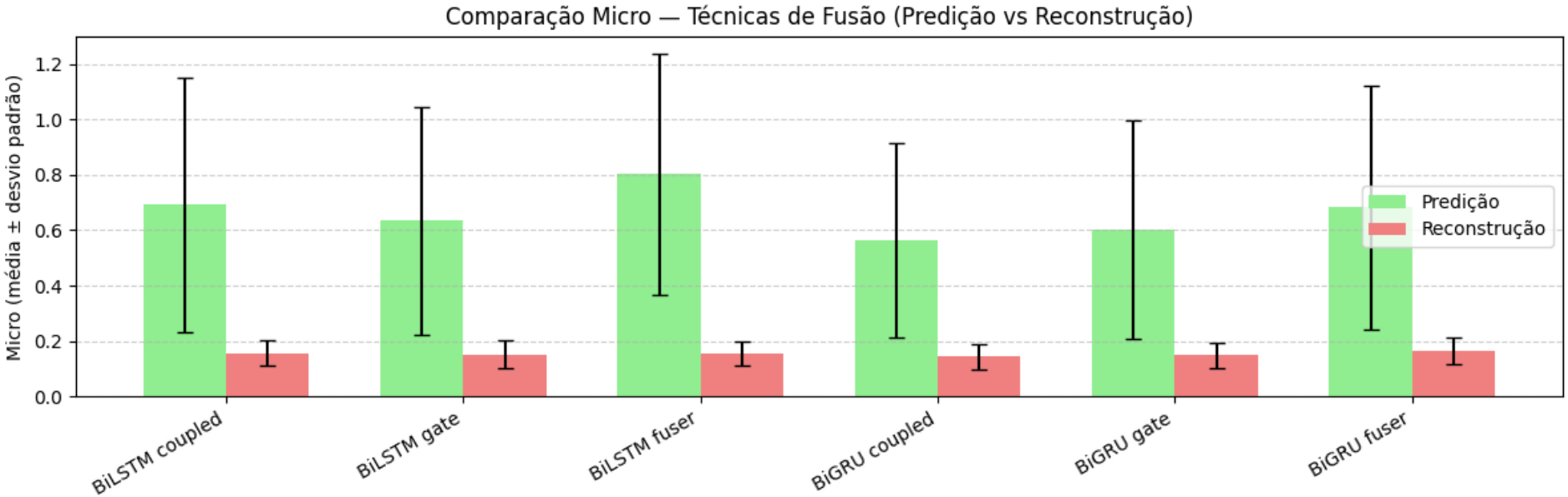
Resultados - MSE micro na predição e reconstrução



A reconstrução alcança resultados melhores que a predição, apesar de ser mais complexa, pois além de informar o que a predição já faz, a reconstrução recupera os valores ausentes da série temporal por difusão.

REVER OS VALORES DA MELHOR EPOCA

Resultados - MSE micro na predição e reconstrução/ Técnicas de fusão pós-BiRNN



Predição e Reconstrução (MSE micro)

TABELA ORDENADA PELOS MELHORES RESULTADOS - REGRA DE 2 DESVIOS PADRÃO / COUPLED E NÃO COUPLED - OS TRES TIPOS DE FUSÃO

Aplicação em LNP - HARNN

resumir o

Referências I

[1] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of the National Academy of Sciences*, vol. 79, no. 8, pp. 2554–2558, 1982.

[2] M. I. Jordan, "Attractor dynamics and parallelism in a connectionist sequential machine," in *Proceedings of the 8th Annual Conference of the Cognitive Science Society*, pp. 531–546, 1986.

[3] J. L. Elman, "Finding structure in time," *Cognitive Science*, vol. 14, no. 2, pp. 179–211, 1990.

[4] P. J. Werbos, "Backpropagation through time: What it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

[5] S. Hochreiter, *Untersuchungen zu dynamischen neuronalen Netzen*. PhD thesis, Technische Universität München, 1991.

[6] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[7] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[8] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional lstm networks," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, pp. 2047–2052, 2005.

[9] K. Cho, B. van Merriënboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[10] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[11] D. Serdyuk, N. R. Ke, A. Sordoni, A. Trischler, C. Pal, and Y. Bengio, "Twin networks: Matching the future for sequence generation," in *International Conference on Learning Representations (ICLR)*, 2018. arXiv:1708.06742.

[12] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *Proceedings of the 30th International Conference on Machine Learning (ICML)*, pp. 1310–1318, 2013.

[13] Y. Gal and Z. Ghahramani, "A theoretically grounded application of dropout in recurrent neural networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 1019–1027, 2016. arXiv:1512.05287.

[14] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[15] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 3104–3112, 2014.

[16] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *International Conference on Learning Representations (ICLR)*, 2015. arXiv:1409.0473.

Referências II

[17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 5998–6008, 2017.

[18] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[19] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI Technical Report*, 2019.

[20] T. B. Brown *et al.*, "Language models are few-shot learners," *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[21] R. Bommasani *et al.*, "On the opportunities and risks of foundation models," *arXiv preprint arXiv:2108.07258*, 2021.