RNNs em Séries Temporais: Arquiteturas, Treinamento e Aplicações

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

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

6 de novembro de 2025

Agenda

Linha do Tempo das RNNs — Resumo

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).
1997+	BiLSTM	Combinação: **LSTM + BiRNN**.
2014	Cho et al.	Introduzem **GRU** (simplifica LSTM).
2014+	BiGRU	Combinação: **GRU + BiRNN**.

BiRNN é arquitetura; BiLSTM/BiGRU = (célula) + bidirecionalidade.

Hopfield Network (1982)

Objetivo: memória associativa (rede recorrente *estática*, não temporal).

Atualização (binária, assíncrona):

$$s_i(t+1) \;=\; \text{sign}\left(\sum_j w_{ij}\, s_j(t)\right), \quad s_i \in \{-1,+1\},\; w_{ij} = w_{ji},\; w_{ii} = 0.$$

Energia (Lyapunov):

$$E(\mathrm{s}) \;=\; -\frac{1}{2}\sum_{i\neq j} w_{ij}\, s_i s_j \;-\; \sum_i \theta_i\, s_i$$

(diminui a cada atualização assíncrona, garantindo convergência para atratores).

Observações:

- Não modela sequência temporal x_t; armazena *padrões* como mínimos de energia.
- Versões contínuas usam tanh$(\cdot)$ e dinâmica diferencial (não necessário aqui).

Jordan Network (1986)

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 RNN / SRN (1990)

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*.

RNN como Rede Desenrolada no Tempo

Forward pass (inferência):

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

Rede desenrolada (equivalente feedforward):

$$(x_1 \to h_1 \to y_1),\ (x_2 \to h_2 \to y_2),\ \ldots,\ (x_T \to h_T \to y_T)$$

Perda total:

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

Intuição: A RNN aprende uma **representação dinâmica** da sequência: cada h_t combina a entrada atual (x_t) com a memória do passado (h_{t-1}).

Ponto-chave: A dependência temporal é explícita — o passado afeta o futuro.

Werbos (1990) — Backpropagation Through Time (BPTT)

Objetivo: calcular gradientes levando em conta dependências temporais.

Gradiente do estado oculto:

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

Termo recorrente:

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

Gradiente final para os pesos:

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

Interpretação:

- O erro em t depende do erro no futuro $(t+1, t+2, \dots)$.
- RNN = rede **recursiva no forward** e **recursiva no backward**.

Por que o Gradiente Desaparece / Explode?

Produto recorrente de Jacobianos:

$$\frac{\partial L}{\partial h_t} = \frac{\partial L}{\partial h_T} \prod_{k=t+1}^{T} (D_k W_{hh})$$

Se

$$\|W_{hh}\|_2 < 1 \quad \Rightarrow \quad \text{gradiente} \to 0 \text{ (vanishing)}$$

Se

$$\|W_{hh}\|_2 > 1 \quad \Rightarrow \quad \text{gradiente} \to \infty \text{ (exploding)}$$

Consequência prática:

- RNN simples **não aprende dependências longas**.
- Treinamento pode travar (sem aprendizado) ou divergir.

Motivação para LSTM (1997) e GRU (2014): Criar **caminhos de gradiente estáveis** ao longo do tempo.

Hochreiter (1991) — Vanishing Gradient

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

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:

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

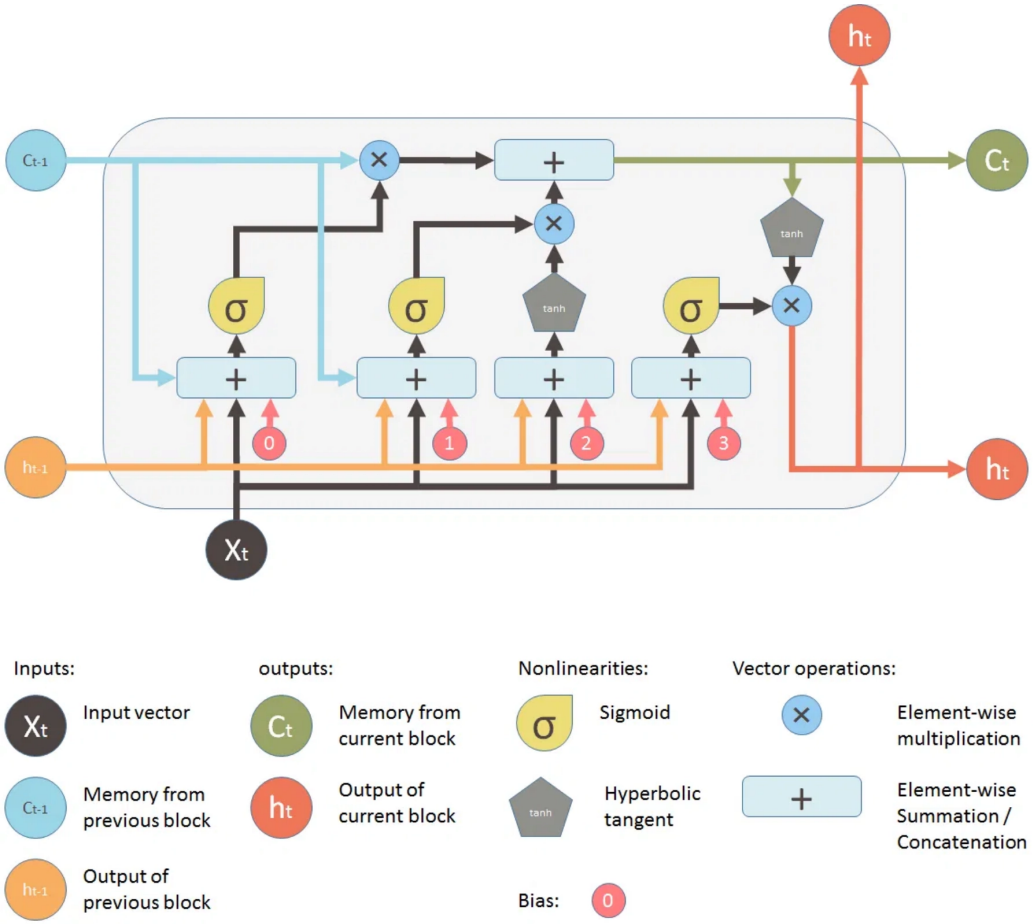
Por que isso evita vanishing gradient?

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

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.)



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)

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)

BiLSTM — Long Short-Term Memory Bidirecional

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:

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

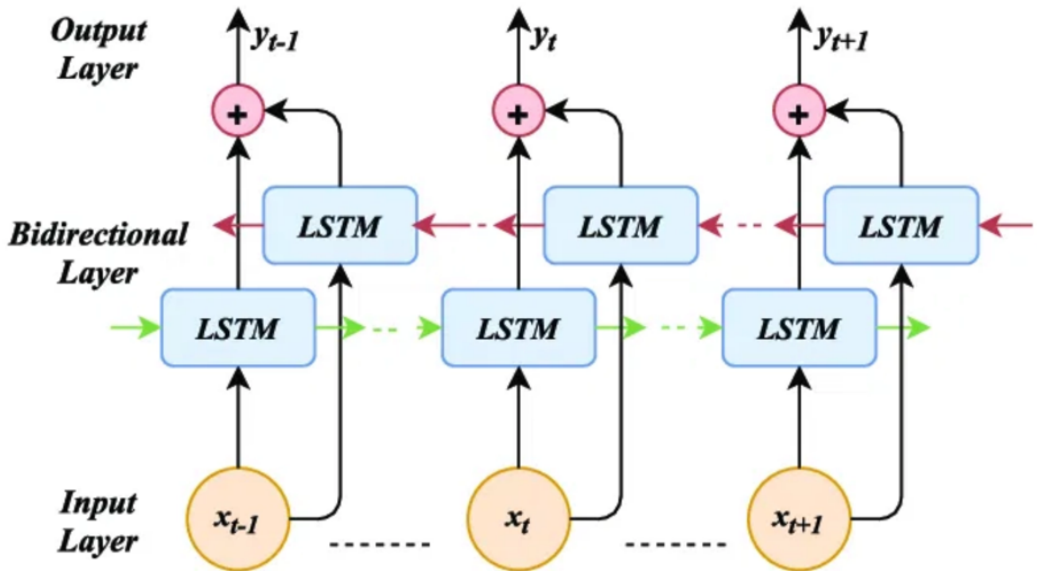
Fusão (saída no tempo t):

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

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.)



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)

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:

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

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

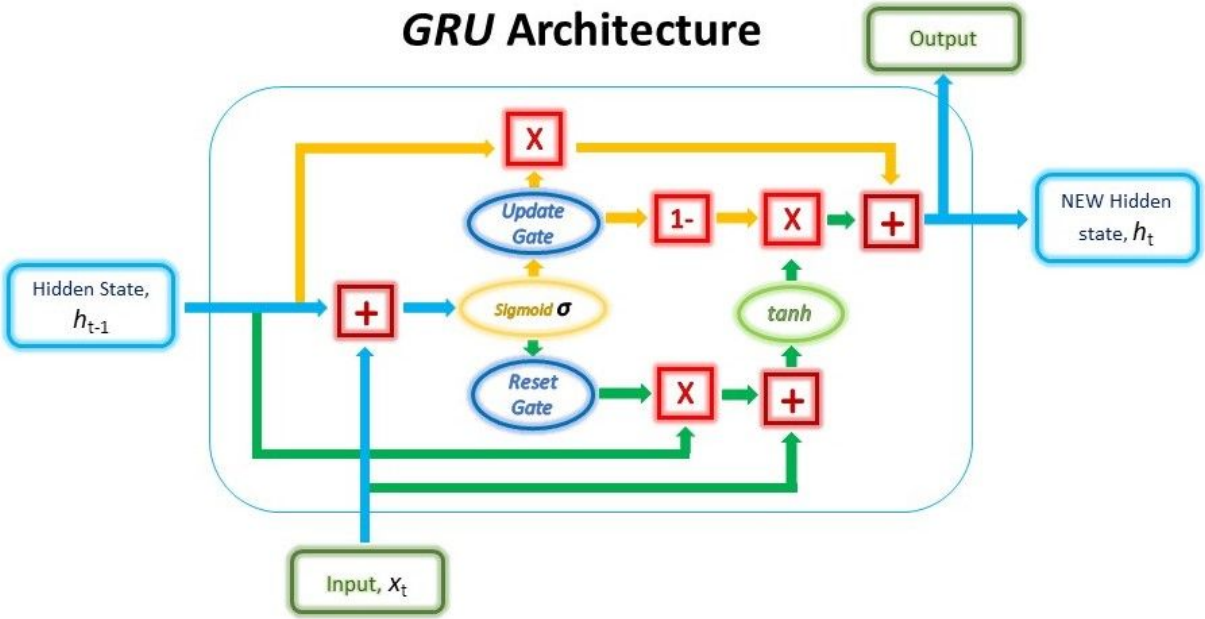
$$\tilde{h}_t = \tanh(W_{xh}x_t + W_{hh}(r_t \odot h_{t-1}) + b_h) \quad \text{(conteúdo candidato)}$$

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

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.)



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).

BiGRU — Gated Recurrent Unit Bidirecional

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):

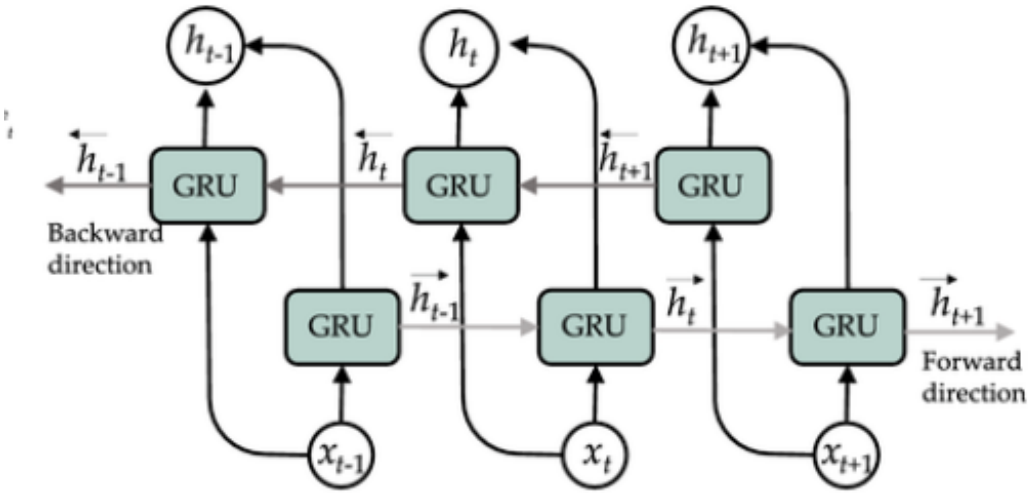
$$h_t = [\,\overrightarrow{h}_t\,;\,\overleftarrow{h}_t\,] \quad \text{(concatenação)}$$

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.

BiGRU — Gated Recurrent Unit Bidirecional (cont.)

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")

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.***

Fusão Bidirecional pós-BiRNN: Concatenação direta

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)

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

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

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*)

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

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

- 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 $\rightarrow$ causal.

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

Hoje: Quando Usar RNN, LSTM, GRU ou Transformers?

RNN / LSTM / GRU ainda são vantajosas quando:

- Dados são **sequenciais e contínuos** (sensores, IIoT, sinais biomédicos).
- O sistema precisa ser **causal** (previsão só com passado).
- Ambiente é embarcado / edge $\rightarrow$ **memória limitada**.
- Latência muito baixa importa.

Transformers são preferíveis quando:

- Dependências são **não locais** e densas (NLP, visão, áudio).
- Disponibilidade de GPU é alta.
- Dataset grande $\rightarrow$ atenção captura relações globais.

Resumo: RNNs **não desapareceram** — elas foram **especializadas** para cenários industriais e temporais.

Comparação entre LSTM, GRU, BiLSTM e BiGRU

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.

Regularização e Estabilização em RNNs

Dropout e Layer Normalization são técnicas cruciais para melhorar a **generalização** e a **estabilidade numérica** durante o treinamento de RNNs.

1. Dropout em RNNs

- Introduz aleatoriamente zeros em frações fixas das ativações entre camadas e/ou ao longo da sequência.
- Evita coadaptação excessiva entre neurônios e reduz overfitting.
- Em RNNs, é aplicado de forma **consistente no tempo** — o mesmo *mask* é usado em todas as etapas da sequência (variational dropout). Isso evita que o ruído varie entre timesteps, preservando a coerência temporal.
- 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

Regularização e Estabilização em RNNs (cont.)

2. Layer Normalization (LN)

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

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

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:

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

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.

Resumo: Dropout e LayerNorm em RNNs

Resumo:

- *Dropout* $\rightarrow$ melhora a **robustez e generalização**.
- *LayerNorm* $\rightarrow$ estabiliza o fluxo de gradiente e acelera o aprendizado.
- Uso combinado: essencial em RNNs profundas ou bidirecionais para evitar explosão/desvanecimento do gradiente.

Integração de Dropout e LayerNorm em BiRNNs

Dropout e **Layer Normalization (LN)** atuam de forma complementar para estabilizar e regularizar redes recorrentes bidirecionais (BiRNN / BiGRU).

Fluxo de processamento em cada passo de tempo t:

$$\begin{aligned} h_t^{\rightarrow} &= \mathrm{GRU}_f(x_t, h_{t-1}^{\rightarrow}) \\ h_t^{\leftarrow} &= \mathrm{GRU}_b(x_t, h_{t+1}^{\leftarrow}) \\ \tilde{h}_t &= \mathrm{LN}\left([h_t^{\rightarrow}; h_t^{\leftarrow}]\right) \\ \hat{h}_t &= d \odot \tilde{h}_t \end{aligned}$$

$h_t^{\rightarrow}, h_t^{\leftarrow}$: estados forward e backward

$[h_t^{\rightarrow}; h_t^{\leftarrow}]$: concatenação bidirecional

$\mathrm{LN}(\cdot)$: LayerNorm, normaliza ativações por amostra

$d \sim \mathrm{Bernoulli}(p)$: máscara de dropout (fixa na sequência)

$\odot$: produto elemento a elemento

Intuição prática:

- **LayerNorm** estabiliza as escalas de ativação entre as duas direções, evitando saturação de portões e explosões de gradiente.
- **Dropout** (variational) atua depois da normalização para reduzir overfitting, sem destruir a coerência temporal da sequência.
- A combinação LN $\rightarrow$ Dropout permite treinar BiRNNs mais profundas e mais gerais, especialmente em sinais ruidosos do mundo real.

Cenário 1: Geração de Séries Temporais

Objetivo: aprender a distribuição dos dados temporais e gerar novas sequências realistas.

$$p(x_1, x_2, \ldots, x_T) = \prod_{t=1}^{T} p(x_t \mid x_{<t}) \quad \text{onde} \quad x_{<t} = (x_1, \ldots, x_{t-1})$$

- Treina-se a rede para maximizar a log-verossimilhança:

$$\mathcal{L} = \sum_{t=1}^{T} \log p(x_t \mid h_t)$$

- Amostragem autoregressiva: gerar x_1, atualizar h_1, gerar $x_2 \sim p(x_2 \mid h_1)$, etc.

Aplicação: simulação de sinais de sensores, séries financeiras sintéticas, dados para teste de robustez.

Cenário 2: Reconstrução de Séries Temporais

Problema: temos série temporal com **dados ausentes**.

- Série parcial: $X = \{x_1, x_2, \ldots, x_T\}$
- Máscara de observação: $M = \{m_t \in \{0,1\}\}_{t=1}^{T}$ com $m_t = 1$ se o valor é conhecido e $m_t = 0$ se está faltante.

Loss com máscara (MSE apenas nos pontos conhecidos):

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

Estratégias:

- **BiLSTM forward/backward**: usa contexto passado e futuro para imputar $\hat{x}_t$.
- **Autoencoder temporal**: um encoder gera um embedding z da série parcial; um decoder reconstrói toda a série.

Aplicações:

- Recuperação de sensores offshore / IoT,
- preenchimento de lacunas em dados médicos,
- reconstrução de históricos de equipamentos industriais críticos.

Modelos Avaliados

- **LSTM**
  3 portas (entrada, esquecimento, saída), estado de célula explícito.
- **GRU**
  2 portas (reset, update), menos parâmetros, mais rápido.
- **BiLSTM / BiGRU**
  Processamento bidirecional: passado e futuro ao mesmo tempo.

Cenários de teste:

1. Geração de séries temporais.
2. Reconstrução de séries faltantes.

Metodologia de Treinamento e Avaliação

Figuras de Mérito:

- MSE (Mean Squared Error)
- MAE (Mean Absolute Error)
- RMSE (Root Mean Squared Error)
- R^2 (Coeficiente de Determinação)

Setup de Treino:

- Divisão de dados: 70% treino, 15% validação, 15% teste.
- Validação cruzada: k-fold, $k = 5$.
- Otimizador: Adam (+ scheduler de taxa de aprendizado).
- Early stopping com paciência de $\sim$20 épocas.

Objetivo final:

- Baixo erro de reconstrução/previsão em *test*.
- Estabilidade (sem explosão) em cenários com ruído.
- Robustez para sensores industriais do mundo real.

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.

Referências Clássicas

- Jordan, M. I. (1986). Attractor dynamics and parallelism in a connectionist sequential machine.
- Elman, J. L. (1990). Finding structure in time. Cognitive Science, 14(2), 179–211.
- Werbos, P. J. (1990). Backpropagation through time: what it does and how to do it. Proc. IEEE.
- Hochreiter, S., & Schmidhuber, J. (1997). *Long short-term memory*. Neural Computation, 9(8), 1735–1780.
- Cho, K. et al. (2014). *Learning phrase representations using RNN encoder-decoder for statistical machine translation*. arXiv:1406.1078.
- Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). *Empirical evaluation of gated recurrent neural networks on sequence modeling*. arXiv:1412.3555.
- Schuster, M., & Paliwal, K. K. (1997). *Bidirectional recurrent neural networks*. IEEE Trans. Signal Processing, 45(11), 2673–2681.
- Bengio, Y., Simard, P., & Frasconi, P. (1994). *Learning long-term dependencies with gradient descent is difficult*. IEEE Trans. Neural Networks, 5(2), 157–166.

Referências Recentes / Aplicações

- Liguori, A. et al. (2021). Reconstrução de séries ambientais internas com autoencoders. *Building and Environment*, 191, 107588.
- Guijo-Rubio, D., Gutiérrez, P. A., & Hervás-Martínez, C. (2023). Reconstrução de séries espaço-temporais de altura de onda. *Applied Soft Computing*, 143, 110417.
- Rasul, K. et al. (2021). *Autoregressive denoising diffusion models for multivariate probabilistic time series forecasting*. ICML.
- Yuan, X., Qiao, L., & Chen, C. (2024). *Diffusion-TS: Interpretable diffusion for general time series generation*. arXiv:2403.01742.
- Lin, L. et al. (2024). *Diffusion models for time-series applications: a survey*. Frontiers of IT & Electronic Engineering, 25, 1–23.
- Yunita, A. et al. (2025). *A comparative study of RNN, LSTM, GRU, and hybrid models*. MethodsX, 13, 103073.