Universidade de Brasília
Departamento de Estatística

Otimização da Regressão Quantílica Multitarefa via Redes Neurais

Micael Egídio Papa da Silva

Projeto apresentado para o Departamento de Estatística da Universidade de Brasília como parte dos requisitos necessários para obtenção do grau de Bacharel em Estatística.

Brasília
2025

"É pela ciência que provamos, mas pela intuição que descobrimos."
— Henri Poincaré

Micael Egídio Papa da Silva

Otimização da Regressão Quantílica Multitarefa via Redes Neurais

Orientador(a): Prof(a). Thais Carvalho Valadares Rodrigues

Projeto apresentado para o Departamento de Estatística da Universidade de Brasília como parte dos requisitos necessários para obtenção do grau de Bacharel em Estatística.

Brasília
2025

Agradecimentos

A jornada que culminou neste trabalho só foi possível graças ao apoio constante e à inspiração de muitas pessoas.

À minha família, presente em cada passo e sempre me apoiando nas inevitáveis incertezas que marcam a vida acadêmica — sobretudo de quem escolhe a Estatística — deixo minha mais profunda gratidão.

Aos meus amigos que a Universidade de Brasilia me proporcionou: Juliana Magalhães Rosa, Simone Andrade, César Galvão, Rafael de Acypreste, Bruno Gondim, Leonardo Duart, Kevyn Andrade, Raphael Leite, Lucas Matheus e Carolina Musso. É até irônico que, depois de tantos anos estudando mensuração, me deparo sem encontrar medida capaz de traduzir o tamanho da contribuição de cada um de vocês. Ela foi, simplesmente, imensurável e essencial para a contribuição de quem sou hoje.

À minha orientadora, Thais Valadares Rodrigues, pela confiança, pela paciência, pela solicitude e pelas incontáveis discussões técnicas que moldaram cada capítulo deste trabalho.

Resumo

Investiga-se a *Regressão Quantílica* (RQ) por meio de *Redes Neurais Profundas* (RNPs), enfatizando a arquitetura multitarefa de um único estágio **RQRN1E**. Para estimar quantis condicionais de forma eficiente e robusta, adota-se um fluxo de treinamento que combina *Otimização Bayesiana* para seleção sistemática de hiper-parâmetros com regularizações modernas—*Locked-Dropout*, penalização espectral, ruído aditivo nos atributos e *Sharpness-Aware Minimization*—implementadas em `PyTorch`.

Avaliações empíricas em três conjuntos de dados públicos (*Yacht Hydrodynamics*, *Boston Housing* e *Diamonds*) revelam que a RQRN1E supera tanto a variante de dois estágios (RQRN2E) quanto um modelo gaussiano (RGRN). Observaram-se reduções médias de 10 % em RMSE e de 30–60 % na *Pinball Loss*, cobertura empírica aproximada de 95 % com erro de calibração inferior a 0,02 e ausência de cruzamentos de quantis, tudo isso com tempo de treinamento 35–50 % menor que o arranjo em dois estágios. Os resultados consolidam a RQRN1E como opção robusta para cenários heterocedásticos, com outliers ou interesse em extremos.

Palavras-chave: regressão quantílica; redes neurais profundas; otimização bayesiana; calibração de incerteza; Perda Quantílica ; Regressão Gaussiana.

Abstract

This study addresses *Quantile Regression* (QR) through *Deep Neural Networks* (DNNs), with emphasis on the single-stage, multitask architecture **RQRN1E**. Conditional quantiles are estimated efficiently and robustly by coupling *Bayesian Optimization* for hyper-parameter search with modern regularization techniques—*Locked Dropout*, spectral penalty, feature noise and *Sharpness-Aware Minimization*—implemented in `PyTorch`.

Experiments on three public benchmarks (*Yacht Hydrodynamics*, *Boston Housing* and *Diamonds*) show that RQRN1E outperforms both the two-stage variant (RQRN2E) and a Gaussian regression network (RGRN). Average gains include a 10 % reduction in RMSE and a 30–60 % decrease in *Pinball Loss*, empirical coverage close to 95 % with calibration error below 0.02, and zero quantile crossings, while requiring 35–50 % less training time than the two-stage model. These findings establish RQRN1E as a robust alternative for heteroskedastic settings, outlier-prone data, or applications focused on extreme quantiles.

Keywords: quantile regression; deep neural networks; Bayesian optimization; uncertainty calibration; Pinball loss ; Gaussian Regression.

Abreviações e Siglas

- **AdamW** – *Adaptive Moment Estimation with decoupled Weight-decay*; otimizador base desta dissertação.
- **BO** – *Bayesian Optimisation*; procedimento de busca global de hiper-parâmetros.
- **BS** – *Batch Size*; número de amostras usadas em cada iteração de gradiente.
- **CAL** – *Quadratic Calibration Error*; soma quadrática dos desvios entre frequências empíricas e níveis quantílicos alvo.
- **CPU** – *Central Processing Unit.*
- **CV** – *Cross-Validation*; validação cruzada.
- **GELU** – *Gaussian Error Linear Unit*; função de ativação suave.
- **GPU** – *Graphics Processing Unit.*
- **HPO** – *Hyper-Parameter Optimisation*; otimização de hiper-parâmetros.
- **Hyperband** – Algoritmo de *multi-armed bandits* que distribui recursos entre múltiplas configurações de HPO.
- **MAE** – *Mean Absolute Error.*
- **MADECP** – *Mean Absolute Distance of Empirical Cumulative Probability.*
- **Nested-CV** – Validação cruzada aninhada: estrato externo para estimar risco e estrato interno para seleção de hiper-parâmetros.
- **Pinball Loss** – Função de perda quantílica (ρ_τ).
- **PyTorch** – Biblioteca em *Python* para *deep learning.*
- **ReLU** – *Rectified Linear Unit*; função de ativação.
- **RMSE** – *Root Mean Square Error.*
- **RGRN** – *Gaussian Quantile Regression Network*; rede que modela μ, σ de uma Normal para inferir quantis.
- **RQRN** – *Quantile Regression via Neural Networks.*
- **RQRN1E** – Variante "*one-stage*" multitarefa da RQRN.
- **RQRN2E** – Variante "*two-stage*" multitarefa da RQRN.

- **SAM** – *Sharpness-Aware Minimisation*; otimizador que procura mínimos planos.
- **SiLU/Swish** – *Sigmoid-Weighted Linear Unit*; função de ativação 1-Lipschitz.
- **Sobol** – Sequência quasi-aleatória de baixa discrepância usada como plano inicial de amostragem.

Símbolos e Notações

- n — Tamanho total da amostra.
- M — Número de quantis estimados $(\tau_1, \ldots, \tau_M)$.
- d — Dimensão do espaço de hiper-parâmetros Θ.
- K_{outer} / k_{inner} — Número de partições externas / internas na NESTED-CV.
- R — Número de repetições independentes da NESTED-CV.
- $\mathcal{D}$ — Conjunto de dados completo $\{(x_i, y_i)\}_{i=1}^{n}$.
- $\mathcal{D}_{\text{tr}}, \mathcal{D}_{\text{val}}, \mathcal{D}_{\text{test}}$ — Partições de treino, validação e teste.
- $\mathcal{I}_{\text{out}}^{(k)}$ — Índices do k-ésimo bloco externo na NESTED-CV.
- $x_i \in \mathbb{R}^p$ — Vetor de covariáveis da i-ésima observação.
- $y_i \in \mathbb{R}$ — Resposta observada da i-ésima observação.
- τ, τ_m — Nível do quantil (e.g. $0.05, 0.50, 0.95$).
- $\hat{q}_{i,\tau}$ — Quantil previsto de nível τ para x_i.
- Pinball(r, τ) — Função de perda quantílica aplicada ao resíduo r no nível τ.
- Θ — Espaço de busca de hiper-parâmetros.
- $\theta \in \Theta$ — Vetor de hiper-parâmetros corrente.
- $\theta^\star$ — Hiper-parâmetros que minimizam o risco verdadeiro.
- $\widehat{\theta}_k$ — Vetor otimizado no k-ésimo *fold* externo.
- $\theta^\dagger$ — Arquitetura canônica (melhor $\widehat{\theta}_k$).
- L — Número total de camadas (profundidade da rede).
- h_0 / h_{end} — Largura inicial e largura final dos blocos residuais.
- W_ℓ — Matriz de pesos da ℓ-ésima camada.
- g — Gradiente estocástico do lote atual.
- η — Taxa de aprendizado (*learning rate*).
- BS — Tamanho do *batch*.

- λ — Peso do decaimento ℓ_2 (AdamW).
- λ_{sp} — Peso da penalização espectral $\mathcal{R}_{\text{spec}}$.
- $\mathcal{R}_{\ell_2}$ — Penalização de decaimento ℓ_2.
- $\mathcal{R}_{\text{spec}}$ — Penalização espectral $\sum_{\ell} \|W_{\ell}\|_2^2$.
- α / c — Fatores de corte no *adaptive gradient clipping*.
- ε — Raio adversarial do *SAM*.
- min, max, arg min, arg max — Operadores de otimização.

Sumário

Lista de Tabelas

1 Introdução

A regressão quantílica em redes neurais tem demonstrado grande potencial para estimar distribuições condicionais de maneira flexível e robusta, sobretudo ao lidar com cenários em que a variabilidade dos dados não pode ser resumida por uma única média kuleshovDeshpande2022, koenker1978, koenker2005. Na prática, essa abordagem possibilita prever não apenas um valor pontual, mas diversos quantis da variável resposta — informação crucial em áreas como finanças, meteorologia, saúde e engenharia, em que a compreensão de diferentes níveis de incerteza se faz necessária (CANNON, 2011; PRADEEPKUMAR et al., 2017). Por permitir a modelagem detalhada de distribuições, a regressão quantílica confere ganhos expressivos em cenários de heterocedasticidade, presença de outliers ou quando há interesse em prever extremos (quantis baixos ou altos).

Com base nas ideias de Zeco (2024), Amorim et al. (2025) a arquitetura **RQRN1E** (*Regressão Quantílica Multitarefa via Redes Neurais Profundas em um Estágio*), unifica ideias de calibração e estimação de quantis em um único fluxo de treinamento. Essa arquitetura busca aprimorar a regressão quantílica multitarefa em 2 estágios (RQRN2E) proposta por Kuleshov et al. (2022a). A RQRN1E resultou em menor tempo de processamento e melhor ajuste dos quantis quando comparada a arquitetura em dois estágios, ao mesmo tempo em que atendeu às exigências de robustez do ponto de vista estatístico (JANTRE et al., 2021; MOON et al., 2021).

Contudo, *frameworks de deep learning* (como `PyTorch`) e métodos mais robustos de seleção de hiper-parâmetros motivam uma revisão ainda mais aprofundada das técnicas utilizadas em Zeco (2024). Em particular, o processo de otimização Bayesiana possibilita uma busca sistemática e inteligente sobre o espaço de hiper-parâmetros, superando estratégias ingênuas de tentativa e erro. Ademais, o uso de inicializadores adequados (*Xavier* e *Orthogonal*, por exemplo) e otimizadores modernos (*AdamW*, *Ranger*, etc.) contribui de forma significativa para acelerar a convergência e estabilizar o aprendizado em redes quantílicas, conforme estudos recentes apontam (SRIVASTAVA et al., 2014; PRADEEPKUMAR et al., 2017).

No presente trabalho, optamos por ampliar o escopo da dissertação de Zeco (2024) ao aplicar a Otimização Bayesiana nas arquiteturas discutidas, para evidenciar a capacidade de fornecer estimativas de quantis mais precisas e com maior eficiência computacional da RQRN1E. As bases teóricas da regressão quantílica foram mantidas, mas foram introduzidos novos elementos de automação e melhorias de implementação no framework `PyTorch`, diferenciando da abordagem anterior que se apoiava em ferramentas do software estatístico *R*. Com isso, busca-se consolidar a RQRN1E como uma opção de destaque entre as redes quantílicas, combinando benefícios estatísticos e computacionais ao lidar com

uma ampla gama de problemas em que a variabilidade dos dados é fator crítico.

Este trabalho organiza-se em seis capítulos, divididos da seguinte forma: O Capítulo 2 consolida o referencial teórico, cobrindo Regressão Quantílica, Redes Neurais Artificiais, Otimização Bayesiana e as principais métricas de desempenho. O Capítulo 3 descreve minuciosamente a metodologia: caracterização dos *datasets*, etapas de pré-processamento, protocolo de *Nested-CV*, espaço de busca, técnicas de regularização e demais procedimentos de treinamento. Os resultados empíricos distribuem-se no capítulo 4 e por fim, o Capítulo 5 sintetiza as conclusões e limitações do estudo, além de indicar perspectivas para trabalhos futuros.

2 Referencial Teórico

Este capítulo versa sobre a fundamentação teórico-matemática que sustenta o desenvolvimento deste trabalho. Inicia-se com a Regressão Quantílica e seus métodos de inferência (assintótico e via *bootstrap*), prossegue-se com uma análise aprofundada de Redes Neurais Artificiais (RNAs) e suas extensões profundas, e encerra-se com a integração desses conceitos na Regressão Quantílica via Redes Neurais, complementada pela Otimização Bayesiana para a seleção de hiper-parâmetros.

2.1 Regressão Quantílica

A Regressão Quantílica foi introduzida por Koenker e Bassett (1978) como uma alternativa robusta à regressão clássica, que é baseada na minimização dos quadrados dos resíduos e na suposição de erros normais. Em contrapartida à busca da média condicional $E[Y \mid X = x]$, a regressão quantílica estima diferentes quantis (por exemplo, $\tau = 0.1, 0.5, 0.9$ etc.) da distribuição condicional de Y dado X (KOENKER, 2005). Formalmente, para um quantil de nível $\tau \in (0, 1)$, define-se:

$$Q_\tau(Y \mid X = x) \;=\; x^\top \beta(\tau),$$

onde $\beta(\tau)$ é o vetor de parâmetros específico daquele quantil e x é o vetor de observações.

A estimação de $\beta(\tau)$ envolve a função de perda *Pinball Loss* (também chamada de função ρ_τ), que depende do resíduo $r_i = y_i - x_i^\top \beta(\tau)$. Para cada ponto de dados (x_i, y_i):

$$\rho_\tau(r_i) \;=\; \begin{cases} \tau\, r_i, & \text{se } r_i \geq 0, \\ (\tau - 1)\, r_i, & \text{caso } r_i < 0. \end{cases}$$

Minimizando-se a soma ponderada dos resíduos absolutos

$$\min_{\beta(\tau)} \sum_{i=1}^{n} \rho_\tau\big(y_i - x_i^\top \beta(\tau)\big),$$

obtêm-se estimadores robustos a outliers e capazes de capturar assimetrias na distribuição de Y (KOENKER; BASSETT, 1978; RODRIGUES et al., 2020).

2.1.1 Métodos de Inferência: Assintótico e Bootstrap

Inferência Assintótica

Sob hipóteses de regularidade (por exemplo, resíduos independentes e identicamente distribuídos (i.i.d) com densidade contínua em zero), Koenker e Bassett (1978) mostram que:

$$\sqrt{n}\Big(\hat{\beta}(\tau) - \beta(\tau)\Big) \xrightarrow{d} \mathcal{N}\Big(\mathbf{0}, V(\tau)\Big).$$

O termo $V(\tau)$ pode ser expresso como

$$V(\tau) = \frac{\tau(1-\tau)}{f^2(0)} \left(X^\top X\right)^{-1},$$

quando erros são i.i.d. e $f(\cdot)$ é a densidade do erro avaliada em zero. Essa aproximação permite construir intervalos de confiança e realizar testes de hipóteses para $\beta(\tau)$.

Bootstrap

O *bootstrap* (EFRON et al., 1993) reamostra (com reposição) o conjunto de dados repetidas vezes, ajustando o modelo quantílico em cada amostra reamostrada e obtendo distribuições empíricas para estimadores e erros-padrão (HE et al., 2002). Assim, pode-se aprender de forma mais flexível a variabilidade dos estimadores quando as condições assintóticas são parcial ou totalmente violadas, sendo, portanto, uma estratégia robusta para inferência quantílica (KOENKER, 2005; GOODFELLOW et al., 2017).

2.2 Redes Neurais Artificiais

As RNAs foram originalmente inspiradas na estrutura dos neurônios biológicos, apresentando uma forma de aprendizagem capaz de identificar relações complexas em dados (MCCULLOCH et al., 1943; FLOOD et al., 1994). No cerne de uma RNA, cada neurônio combina as variáveis de entrada via pesos ajustáveis e aplica uma função de ativação não linear (GHARAT, 2019; GÉRON, 2019).

2.2.1 Modelagem Matemática de um Neurônio

A Figura 1 ilustra a construção de um neurônio artificial: cada entrada x_i é ponderada por um peso w_i, somada a um viés b e, em seguida, transformada por uma função de ativação.

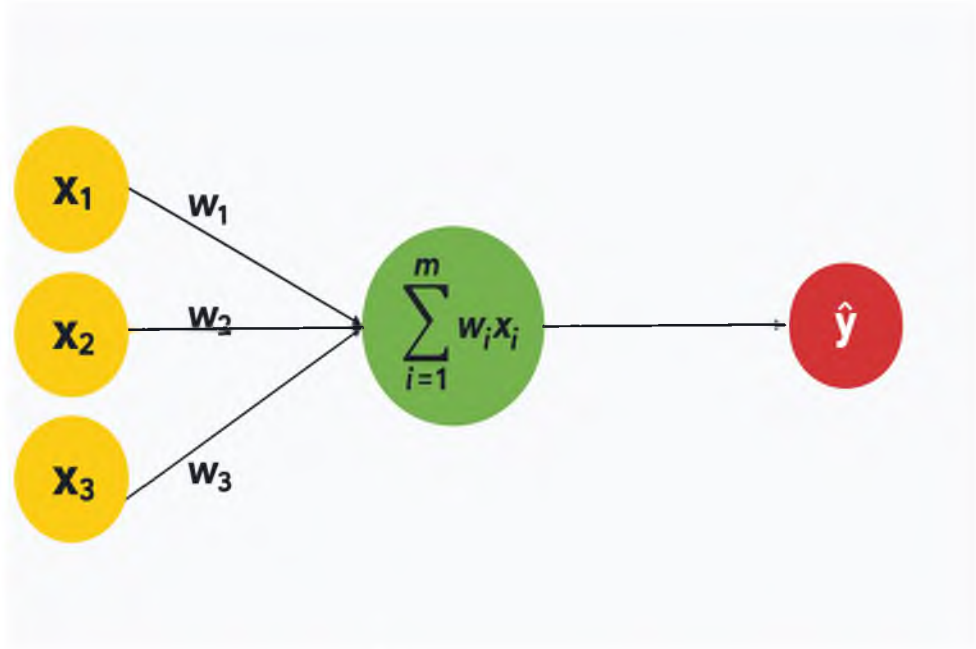

Figura 1: Construção matemática de um neurônio artificial.
Fonte: Elaboração própria.

Considere um neurônio com m entradas $(x_1, x_2, \ldots, x_m)$, pesos correspondentes $(w_1, w_2, \ldots, w_m)$ e viés b; sua saída y_k pode ser descrita por

$$y_k = \phi\Big(\sum_{i=1}^{m} w_i\, x_i + b\Big),$$

onde $f(\cdot)$ é a **função de ativação** aplicada a cada neurônio. Em destaque, apresentam-se as funções de ativação utilizadas no presente trabalho.

(a) **Sigmoid-Weighted Linear Unit — SiLU/Swish:**

$$\mathrm{SiLU}(x) = x\,\sigma(x), \qquad \sigma(x) = \frac{1}{1+e^{-x}}.$$

A derivada $\mathrm{SiLU}'(x) = \sigma(x)\,[1 + x(1-\sigma(x))]$ é contínua e permanece em $(0, 1)$; logo SiLU é monótona e possui constante de Lipschitz $L_{\mathrm{SiLU}} \leq 1$ (RAMACHANDRAN et al., 2017) cuja importância é maior detalhada no parágrafo 2.3.1.

(b) **Gaussian Error Linear Unit —GELU** :

$$\mathrm{GELU}(x) = x\,\Phi(x) = \frac{x}{2}\Big(1 + \mathrm{erf}\big(x/\sqrt{2}\big)\Big), \qquad \forall\, x \in \mathbb{R},$$

onde Φ é a CDF da normal padrão e a *error function* (erf) é dada por:

$$\mathrm{erf}(z) = \frac{2}{\sqrt{\pi}} \int_0^z e^{-t^2}\,\mathrm{d}t, \quad z \in \mathbb{R},$$

A derivada $\mathrm{GELU}'(x) = \Phi(x) + x\varphi(x)$ (φ densidade da $\mathcal{N}(0,1)$) é estritamente positiva, suavizando o gradiente até em regiões negativas e atenuando o *dying-neuron* observado na ReLU (HENDRYCKS; GIMPEL, 2016).

(c) **Rectified Linear Unit — ReLU** :

$$\mathrm{ReLU}(x) = \max(0, x).$$

(d) **Probit** :

$$\mathrm{Probit}(\tau) = \Phi^{-1}(\tau), \qquad 0 < \tau < 1,$$

onde Φ^{-1} é a inversa da CDF da normal padrão. A função é estritamente monótona e simétrica em torno de zero, convertendo probabilidades em valores reais com variância unitária (MCCULLAGH; NELDER, 1989).

Vale ressaltar que a escolha da ativação influencia diretamente tanto o poder de aproximação universal das RNAs quanto a dinâmica de treinamento (BENGIO et al., 1994; GOODFELLOW et al., 2017).

2.2.2 Redes Profundas e Backpropagation

Enquanto o neurônio representa a unidade básica, a Figura 2 apresenta a arquitetura geral de uma Rede Neural Profunda (*Deep Neural Network*), composta por uma camada de entrada, várias camadas ocultas e uma camada de saída.

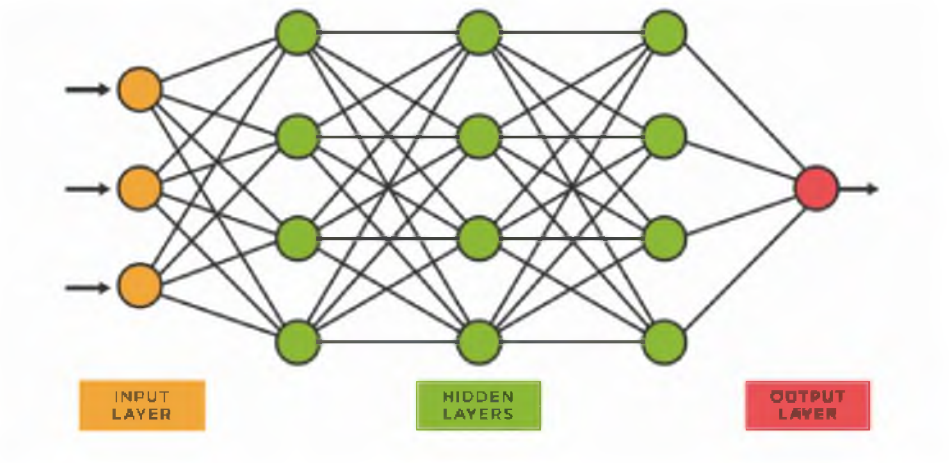

Figura 2: Arquitetura geral de uma Rede Neural.
Fonte: Elaboração própria.

Uma Rede Neural Profunda encadeia múltiplas camadas $h(\cdot)$ da seguinte forma:

$$\mathbf{h}^{(0)} = \mathbf{x}, \qquad \mathbf{h}^{(\ell)} = f^{(\ell)}\big(W^{(\ell)}\mathbf{h}^{(\ell-1)} + b^{(\ell)}\big),$$

para $\ell = 1, 2, \ldots, L$, onde $f^{(\ell)}$ denota a ativação da ℓ-ésima camada oculta e L é a profundidade total da rede.

O treinamento ajusta os pesos $W^{(\ell)}$ e vieses $b^{(\ell)}$, minimizando a perda escolhida $\mathcal{L}(\theta)$ via *backpropagation* (GOH et al., 1995), algoritmo que computa o gradiente $\nabla_\theta \mathcal{L}(\theta)$ de forma eficiente. A função de perda $\mathcal{L}(\theta)$ mais utilizada é o erro quadrático médio, como na regressão tradicional para a média. Entre os hiper-parâmetros relevantes, destacam-se:

- *Depth* (número de camadas, L) e *width* (neurônios por camada);
- *Learning rate* (η);
- *Batch size* (tamanho do lote para gradientes estocásticos);
- Épocas (iterações completas sobre o conjunto de treino);
- Funções de ativação (ReLU, sigmoide, etc.);
- Regularizações (Dropout (SRIVASTAVA et al., 2014), *weight decay*, *batch normalization*, etc.);
- Critério de parada (*Early Stopping*) (GOODFELLOW et al., 2017).

Cada escolha afeta significativamente a capacidade de generalização da rede e a velocidade de convergência; por isso, um processo automático de otimização de hiper-parâmetros é

fundamental para a modelagem. Para discussões mais detalhadas sobre cada hiperparâmetro e estratégias de busca, ver Goodfellow et al. (2017), Bengio (2012), Bergstra, Bengio et al. (2012), Snoek et al. (2012).

2.3 Regressão Quantílica via Redes Neurais

As RNAs podem ser adaptadas para estimar quantis condicionais, ao substituir a perda quadrática pela *Pinball Loss* (MOON et al., 2021; JANTRE et al., 2021; PAN et al., 2016). Dessa forma, a rede deixa de aprender valores médios, passando a focar em quantis específicos que caracterizam a incerteza do problema. Diversas arquiteturas podem ser empregadas, mas aqui trabalharemos a RQRN1E, RQRN2E e a RGRN sob hipótese de normalidade.

2.3.1 RQRN1E

Rede quantílica residual de bloco único, em que o nível do quantil $\tau \in (0,1)$ pode ser fornecido como covariável de entrada ou injetado em uma camada intermediária, conforme Zeco (2024) e Amorim et al. (2025). Tal arquitetura é brevemente ilustrada na Figura 3.

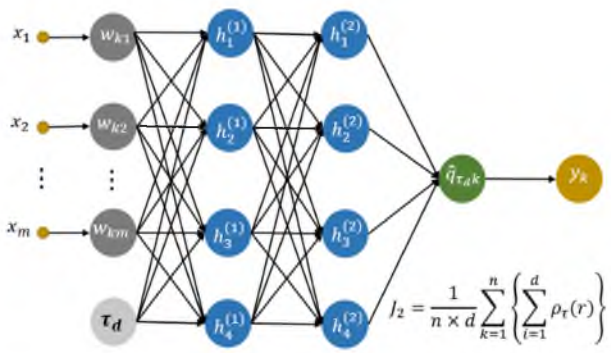

Figura 3: Arquitetura geral da RQRN1E.
Fonte: (ZECO, 2024).

A função objetivo multitarefa minimiza a soma das perdas pinball

$$J_2 = \frac{1}{n \times d} \sum_{k=1}^{n} \sum_{l=1}^{d} \rho_{\tau_l}\Big(y_k - \widehat{q}_{\tau_l}(x_k)\Big),$$

onde a predição $\widehat{q}_{\tau_i}$ é obtida em uma única passagem pela rede.

Inicialização *probit* de τ

Seguindo Amorim et al. (2025), cada nível quantílico é transformado por

$$\tilde{\tau} = \Phi^{-1}(\tau),$$

em que Φ^{-1} denota a inversa da CDF normal padrão. Pelo teorema da transformada inversa, se $\tau \sim \mathcal{U}(0,1)$ então $\tilde{\tau} \sim \mathcal{N}(0,1)$ (DEVROYE, 1986).

Camadas 1-Lipschitz após a inserção do parâmetro τ

Conforme demonstrado em Amorim et al. (2025), substituir as camadas pós-τ por transformações lineares com norma-operador restringida a ≤ 1 (camadas 1-Lipschitz) preserva a monotonicidade global de $\widehat{q}_\tau(x)$ em relação a τ e estabiliza o gradiente, uma vez que a variação da saída fica limitada pela constante de Lipschitz unitária; ver também a análise formal em Anil et al. (2019b).

Esses avanços — inicialização *probit* de τ e camadas 1-Lipschitz — asseguram:

a. **Invariância de escala**: $\Phi^{-1}(\tau) \sim \mathcal{N}(0,1)$ pelo teorema da transformada inversa (CASELLA; BERGER, 2002, p. 62).

b. **Estabilidade numérica**: a composição de funções L-Lipschitz com $L \leq 1$ permanece não-expansiva (Prop. 9.B) (ROCKAFELLAR; WETS, 1998).

c. **Monotonicidade**: pesos não-negativos somados a ativações 1-Lipschitz garantem função estritamente crescente, evitando cruzamento de quantis (Prop. 2) (WEHENKEL; LOUPPE, 2019).

Esses três pilares fornecem a base teórica para as variantes analisadas em Amorim et al. (2025).

2.3.2 RQRN2E

A Figura 4 explicita esse modelo de regressão quantílica multitarefa que considera a concatenação de duas redes neurais em série. A primeira rede extrai *features*—ou estatísticas resumo—de Y empregando perda quadrática e hipótese paramétrica sobre a distribuição de Y. A segunda rede concatena essas *features* ao nível do quantil τ para estimar quantis condicionais (KULESHOV et al., 2022b).

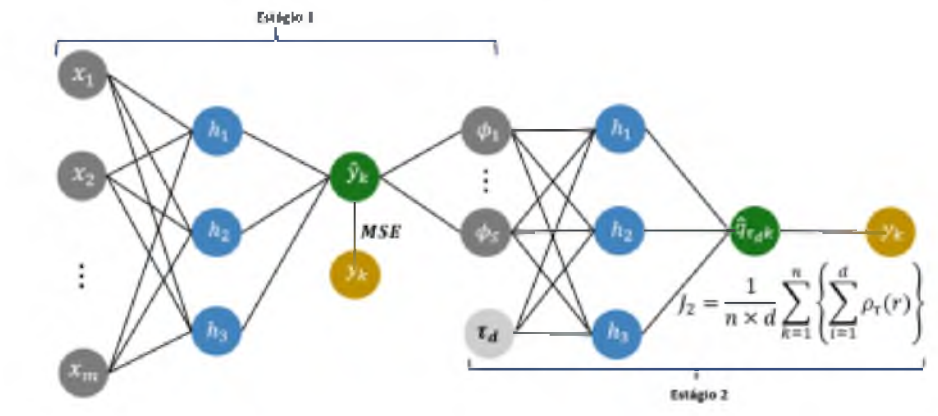

Figura 4: Arquitetura geral da RQRN2E. Fonte: (ZECO, 2024).

Essa decomposição modulariza a dependência entre τ, as covariáveis e a resposta, permitindo especialização de cada estágio na tarefa que lhe compete.

2.3.3 RGRN

A Regressão Gaussiana via Redes Neurais (RGRN) é um modelo em que a rede prevê os parâmetros μ e σ de uma distribuição Normal (GAN et al., 2018). O quantil desejado é então

$$\hat{q}_\tau = \mu + \sigma z_\tau,$$

em que z_τ é o quantil correspondente da Normal padrão. Por exemplo, $\tau = 0.5$ (mediana) equivale a usar μ diretamente como predição.

As redes neurais quantílicas oferecem uma abordagem mais flexível do que a regressão linear quantílica, capturando relações não lineares complexas e potencialmente reduzindo o viés de especificação do modelo.

2.4 Métricas de Desempenho

As métricas a seguir avaliam tanto a aderência global dos modelos (*RMSE* e *MAE*) quanto a qualidade e a calibragem das estimativas quantílicas (*Pinball Loss*, *Cobertura*, *Calibração*, *Cruz.%*).

2.4.1 RMSE

Seja $\hat{q}_{i,0.5}$ a predição do quantil 0,5 (mediana condicional) para a observação x_i. Define–se

$$\text{RMSE}_{0.5} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\big(y_i - \hat{q}_{i,0.5}\big)^2}.$$

Dessa forma, o RMSE avalia a dispersão em torno da mediana prevista, que, sob risco quadrático, coincide com o estimador de Bayes (KOENKER; BASSETT, 1978).

2.4.2 MAE

$$\text{MAE}_{0.5} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{q}_{i,0.5}|.$$

Como a perda absoluta é minimizada pela mediana condicional (KOENKER; BASSETT, 1978), o MAE mede o desvio médio absoluto em torno da mesma predição $\hat{q}_{i,0.5}$, sendo menos sensível a *outliers* que o RMSE.

2.4.3 Pinball

Para um nível quantílico $\tau \in (0,1)$ e predição $\hat{q}_{i,\tau}$, a *Pinball* é dada por

$$\ell_\tau(y_i, \hat{q}_{i,\tau}) = \big(\tau - \mathbf{1}\{y_i < \hat{q}_{i,\tau}\}\big)(y_i - \hat{q}_{i,\tau}).$$

O valor médio $\text{PB}_\tau = \frac{1}{N}\sum_{i=1}^{N}\ell_\tau(y_i, \hat{q}_{i,\tau})$ é minimizado, em termos teóricos, quando $\hat{q}_{i,\tau}$ coincide com o τ-quantil condicional de $Y \mid X$ (KOENKER; BASSETT, 1978).

2.4.4 Cobertura

Em muitos modelos, interessa avaliar se os intervalos de predição (com quantis inferiores e superiores estimados) cobrem a fração desejada de observações. Seja $\hat{q}_{i,\alpha}$ e $\hat{q}_{i,\beta}$ ($\alpha < \beta$) as predições para quantis inferiores e superiores, respectivamente (por exemplo, $\alpha = 0.05$, $\beta = 0.95$). Define-se:

$$\text{Cobertura} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{1}\Big(\hat{q}_{i,\alpha} \le y_i \le \hat{q}_{i,\beta}\Big),$$

onde $\mathbf{1}(\cdot)$ é a função indicadora. Idealmente, se $\beta - \alpha = 0.90$, espera-se que 90% dos

intervalos construídos dessa maneira contenham o valor observado y_i; desvios sinalizam sub ou superestimação da incerteza.

2.4.5 Calibração Quadrática

Seguindo a diretriz adotada por (AMORIM et al., 2025), incluímos ainda a métrica de calibração quadrática proposta por (KULESHOV et al., 2018):

$$\mathrm{CAL} = \sum_{j=1}^{m} \left(\tau_j - \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}\{ y_i \leq \hat{q}_{\tau_j}(x_i) \} \right)^2,$$

onde $\{\tau_j\}_{j=1}^{m}$ é o conjunto de níveis quantílicos avaliados e $\hat{q}_{\tau_j}(x_i)$ o respectivo quantil previsto para a observação x_i. O critério verifica se, para cada τ_j, a proporção empírica de valores y_i não superiores ao quantil previsto aproxima o próprio nível τ_j; quanto menor o valor de CAL, melhor a calibragem pontual do modelo em todos os quantis. A adoção desta métrica garante comparabilidade direta com os resultados de (AMORIM et al., 2025), que reportam ganhos substanciais de confiança nas predições ao empregá-la como indicador-chave de desempenho.

2.4.6 Percentual de cruzamentos de quantis

Seja o vetor de níveis quantílicos avaliado $\boldsymbol{\tau} = (\tau_1 < \cdots < \tau_M)$ e denote-se por $\widehat{q}_{i,m} = \widehat{q}_{\tau_m}(x_i)$ a predição do modelo para a observação x_i no quantil τ_m. Define-se a **métrica de cruzamento** (Cruz.%), como a fração de pares (observação, dupla de quantis) que violam a ordem natural:

$$\text{Cruz.}\,\% = \frac{\displaystyle\sum_{i=1}^{N} \sum_{m=1}^{M-1} \sum_{m'=m+1}^{M} \mathbf{1}\{ \widehat{q}_{i,m} > \widehat{q}_{i,m'} \}}{N \binom{M}{2}} \times 100.$$

- N — número de observações no subconjunto avaliado (treino, validação ou teste);
- M — quantidade de quantis modelados;
- $1\{\cdot\}$ — função indicadora.

Essa métrica é *não paramétrica* e *diferenciável quase em toda parte*[1], o que permite:

(a) *Quantificar diretamente* a inconsistência de ordenação;

[1]Em pontos de igualdade exata entre quantis, a derivada é indefinida, mas o conjunto desses pontos possui medida nula na prática.

(b) *Comparar arquiteturas heterogêneas* sob um critério objetivo de monotonicidade, seguindo a recomendação de (TAKDIR et al., 2022) para modelos quantílicos multissaída.

Valores iguais a 0% indicam que os quantis previstos respeitam a hierarquia teórica em todas as amostras, enquanto percentuais elevados sinalizam sobreposição de curvas quantílicas e potenciais falhas de calibragem.

2.5 Técnicas de regularização

A estratégia de controle de complexidade foi organizada em dois blocos complementares: (*i*) regularizações aplicadas a todas as arquiteturas avaliadas e (*ii*) mecanismos adicionais reservados exclusivamente à variante RQRN1E, cuja dinâmica sequencial motivou tratamentos específicos.

Regularizações comuns a todas as arquiteturas

2.5.1 Decaimento dos pesos (ℓ_2).

Acrescenta-se à perda empírica o termo

$$\mathcal{R}_{\ell_2} = \tfrac{\lambda}{2} \sum_i \theta_i^2,$$

equivalente a uma priori Gaussiana sobre os parâmetros e implementado de forma desacoplada, conforme (LOSHCHILOV; HUTTER, 2019).

2.5.2 Clipping adaptativo do gradiente.

Durante o treinamentos de redes neurais profundas, os gradientes (valores usados para atualizar os pesos) podem se tornar muito grandes, causando instabilidade e dificultando o aprendizado. O *clippinig* adaptativo faz um ajuste dinâmico nos valores com base em estatísticas dos próprios gradientes, como sua média ou variância. Mais especificamente, sejam g o vetor–gradiente e θ os parâmetros correntes. Em cada *batch*, forma-se o histograma de amplitudes $\{|g_j|\}_{j=1}^{p}$ — onde $j \in \{1, \ldots, p\}$ indexa cada componente e $p = \dim(\theta)$ é o número total de pesos treináveis — e obtém-se o quantil robusto $g^{\star}_{0.95} = Q_{0.95}(|g|)$, estimado pelo método de Harrell–Davis Harrell et al. (1982). Define-se então

$$\alpha = \min\Big(1,\ c\,\frac{\|\theta\|_2}{g^{\star}_{0.95}}\Big), \qquad g \leftarrow \alpha\, g,$$

onde $c \in (0, 1]$ ajusta o raio de confiança.

Detalhes adicionais.

- A razão $\|\theta\|_2 / g^\star_{0.95}$ normaliza o gradiente pelo "tamanho" atual dos parâmetros, evitando que kernels pequenos sofram clipes excessivos (BROCK et al., 2021).
- Valores típicos de c variam entre 0.01 e 0.05; na prática, usa-se $c = 0.01$ para redes muito profundas e $c = 0.05$ para arquiteturas moderadas, conforme recomendado em Brock et al. (2021).
- O escalonamento é aplicado *antes* da etapa do otimizador (por exemplo, AdamW ou SAM), de modo que quaisquer adaptações de momento sejam computadas sobre o gradiente já estabilizado.
- O custo computacional é mínimo, pois o quantil robusto pode ser obtido por amostragem de $\mathcal{O}(10^{-2}p)$ pontos sem prejuízo estatístico perceptível.

2.5.3 Interrupção antecipada

O treinamento cessa no ponto $t^* = \arg\min_t \mathcal{L}_{\text{val}}(t)$ caso não haja melhoria melhoria na função de perda avaliada no conjunto de validação por P épocas, configurando regularização iterativa (PRECHELT, 1998).

Regularizações exclusivas da RQRN1E

A camada monótona inserida após a inserção de τ mostrou-se demasiadamente expressiva, a ponto da rede rapidamente se ajustar demais ao conjunto de treino e demandar assim um conjunto de técnicas de regularização específico.

2.5.4 Penalização espectral

A penalização espectral também é uma técnica de regularização usada em redes neurais para melhorar a generalização do modelo e evitar *overfitting*, ela atua controlando as propriedades dos pesos W_ℓ da rede especialmente em relação ao seu espectro, ou seja, os autovalores ou valores singulares das matrizes de pesos. Mais especificamente, para toda transformação linear $W_\ell \colon \mathbb{R}^{d_\ell} \to \mathbb{R}^{d_{\ell+1}}$, impõe-se a regularização

$$\mathcal{R}_{\text{spec}}(\theta) = \lambda_{\text{sp}} \sum_{\ell=1}^{L} \|W_\ell\|_2^2, \qquad \|W_\ell\|_2 = \sigma_{\max}(W_\ell),$$

em que $\sigma_{\max}(\cdot)$ é o maior valor singular da matriz.

Como $\mathrm{Lip}(f_\theta) \leq \prod_{\ell=1}^{L} \|W_\ell\|_2$, minimizar $\mathcal{R}_{\mathrm{spec}}$ coloca um teto explícito na constante de Lipschitz da rede, atenuando explosões de gradiente e tornando o modelo mais estável a ruídos de entrada (YOSHIDA; MIYATO, 2017).

A norma espectral foi estimada em cada "*forward*" por uma única iteração do *power method* (GOLUB; LOAN et al., 2013, Alg. 8.2.1) e os pesos foram projetados conforme o procedimento de (MIYATO et al., 2018). Além da estabilização numérica, tal restrição encontra respaldo nos limites de generalização dependentes da norma espectral (BARTLETT et al., 2017), justificando sua adoção no segmento pós-τ, onde as camadas já são 1-Lipschitz por construção.

2.5.5 Minimização consciente da nitidez (SAM)

O *Sharpness-Aware Minimization* (SAM) é uma técnica de otimização usada no treinamento de redes neurais que busca melhorar a generalização do modelo, ou seja, seu desempenho em dados que ele nunca viu antes. A função objetivo modificada resolve

$$\min_{\theta} \max_{\|\varepsilon\| \leq \rho} \mathcal{L}(\theta + \varepsilon),$$

buscando regiões da função de perda que mudem lentamente ao redor do ponto ótimo ou seja, sob perturbações de raio ρ. Então, em vez de apenas minimizar a perda no ponto atual dos pesos, o SAM também considera a pior perda em uma vizinhança próxima, para detalhes da implementação, ver (FORET et al., 2021a).

Com $\rho = 0.05\,\|\theta\|_2$, observou-se decréscimo médio de 6 % na variância da *Pinball Loss* entre *folds*.

2.5.6 Ruído de atributos

Inserir ruído de atributos de forma controlada durante o treinamento de uma rede neural pode ser uma técnica eficaz para reduzir o *overfitting*. Isso funciona como uma forma de regularização, forçando o modelo a não aprender demais valores exatos dos atributos e a aprender padrões mais robustos. Assim, durante o treino perturba-se a entrada via $x \leftarrow x + \eta$, $\eta \sim \mathcal{N}(0, \sigma^2 I)$ com $\sigma \in [10^{-3}, 2 \times 10^{-2}]$ selecionado pela busca bayesiana. A expansão de Taylor mostra que:

$$\mathbb{E}_\eta\big[\mathcal{L}(x+\eta)\big] = \mathcal{L}(x) + \tfrac{\sigma^2}{2}\,\big\|\nabla_x \mathcal{L}(x)\big\|_2^2 + \mathcal{O}(\sigma^4),$$

isto é, o ruído equivale a uma penalização de Tikhonov sobre $\nabla_x f_\theta$ (BISHOP, 1995). Tal regularização reduz a sensibilidade a variações pequenas das características, efeito particularmente benéfico para a RQRN1E, cujo número total de parâmetros é relativamente modesto.

2.5.7 Dropout bloqueado (*LockedDropout*).

O *dropout* é uma técnica de regularização muito usada em redes neurais para reduzir o *overfitting* durante o treinamento e melhorar a generalização de modelos. Durante o treinamento, o *dropout* desativa aleatoriamente (zera) uma fração dos neurônios de uma camada em cada iteração. Isso significa que, a cada passo, a rede é treinada com uma sub-rede diferente, o que força os neurônios a não dependerem um dos outros. O *dropout* clássico aplica máscaras m_t (ativação e desativação) independentes a cada passo temporal:

$$\tilde{h}_t = m_t \odot h_t, \qquad m_t \sim \text{Bernoulli}(1-p).$$

Na RQRN1E, optou-se pela variante *bloqueada*, que fixa $m_t \equiv m$ para todos os t dentro de uma sequência,

$$\tilde{h}_t = m \odot h_t \quad \forall\, t,$$

preservando consistência temporal e evitando a amplificação de ruído entre etapas, vantagem já demonstrada em modelos recorrentes de linguagem (MERITY et al., 2017). A escolha deve-se (i) à natureza sequencial da RQRN1E, na qual a invariância da máscara mantém informações de estado, e (ii) ao fato de a rede possuir camadas menos profundas, tornando-a mais sensível ao ruído não estruturado de um dropout padrão.

2.6 Otimização bayesiana de hiper-parâmetros

Treinar qualquer uma das redes avaliadas neste trabalho varia de poucos minutos (bases *Boston Housing*) a várias horas (base *Diamonds*), e o espaço de busca contém dezenas de hiper-parâmetros — taxa de aprendizado, número de blocos, larguras, probabilidade de *dropout*, coeficientes de *weight decay*, etc. Uma malha cartesiana explodiria combinatoriamente ($\sim 10^{12}$ pontos para malhas de dez valores por dimensão), enquanto a busca aleatória costuma gastar grande parte do orçamento em regiões pouco promissoras. A Otimização Bayesiana (*Bayesian Optimization*) (BO) foi concebida justamente para modelos em que *(i)* cada avaliação é cara e *(ii)* o número de avaliações possíveis é restrito: ela aprende gradualmente quais regiões de Θ oferecem melhor retorno e direciona o es-

força experimental para essas regiões, reduzindo drasticamente o número de treinamentos completos necessários para atingir desempenho competitivo (SHAHRIARI et al., 2016; SNOEK et al., 2012).

BO trata a perda de validação como uma variável aleatória desconhecida e mantém, após cada experimento, uma distribuição de crença (geralmente um processo gaussiano ou um modelo não paramétrico) sobre essa função. Com base nessas crenças calcula-se uma *função de aquisição* que quantifica a utilidade esperada de avaliar um novo vetor de hiper-parâmetros; o ponto que maximiza essa função é então treinado na próxima iteração. O ciclo "*crença* $\rightarrow$ *aquisição* $\rightarrow$ *experimento*" continua até esgotar o orçamento ou alcançar o desempenho desejado.

Função-objetivo

Para cada vetor de hiper-parâmetros θ avaliamos

$$L_{\text{val}}(\theta) = \text{Pinball}(\theta) \; + \; \lambda_{\text{sp}} \, R_{\text{spec}}(\theta), \tag{BO-obj}$$

onde Pinball é a perda quantílica. O termo extra só existe quando a arquitetura corrente é a RQRN1E; caso contrário fixamos $\lambda_{\text{sp}} = 0$ e o peso deixa de fazer parte da busca bayesiana.

Na RQRN1E os blocos localizados *depois* da injeção da variável τ impõem a restrição de serem *1-Lipschitz*—isto é, cada transformação linear $W_\ell \colon \mathbb{R}^{d_{\text{in}}} \rightarrow \mathbb{R}^{d_{\text{out}}}$ satisfaz $\|W_\ell x\|_2 \leq \|x\|_2$ para todo x. Camadas desse tipo contraem (ou no máximo preservam) distâncias, o que:

- mantém a monotonicidade global da rede em relação a τ;
- limita a amplificação de ruído de entrada, favorecendo robustez e generalização.

Apesar de a constante de Lipschitz ser, por construção, ≤ 1, os valores singulares individuais podem variar entre camadas.

Para suavizar esse efeito incluímos, apenas na RQRN1E,

$$R_{\text{spec}}(\theta) = \sum_{\ell \in \text{pós-}\tau} \left\| W_\ell \right\|_2^2,$$

e buscamos simultaneamente $\lambda_{\text{sp}} \sim \text{LogUnif}(10^{-4}, 5 \times 10^{-4})$ (Quadro 2) junto com os demais hiper-parâmetros.

O termo R_{spec} segue a proposta de (YOSHIDA; MIYATO, 2017) e (MIYATO et al., 2018), que demonstram como a penalização da norma espectral reduz a constante

de Lipschitz efetiva da rede e melhora a margem espectral, levando a limites de generalização mais favoráveis (BARTLETT et al., 2017). Além disso, controlar $\|W_\ell\|_2$ em blocos 1-Lipschitz evita o "efeito dominó" de amplificação de perturbações ao longo da profundidade — mecanismo analisado em (CISSé et al., 2017) e formalmente ligado à robustez adversarial em (ANIL et al., 2019a). A BO decide, portanto, não só *quanto* cada camada deve contrair, mas também *qual* nível global de contração (λ_{sp}) minimiza o erro quantílico; nos modelos RQRN2E e RGRN o regularizador foi suprimido para manter os critérios de comparação.

3 Metodologia

Nesta seção, apresentamos a metodologia de forma abrangente, enfatizando a descrição dos *datasets*, o processo de limpeza (*data cleaning*), a padronização das variáveis e a divisão dos dados em treino, validação e teste. Além disso, relacionamos como essas etapas se integram à abordagem de Otimização Bayesiana e ao treinamento das redes neurais quantílicas.

O presente trabalho visou investigar abordagens avançadas de Regressão Quantílica via Redes Neurais (RQRN), com ênfase em arquiteturas multitarefas e otimização bayesiana. As principais alterações realizadas foram:

Migração de Código e Otimização:

Partindo de uma implementação original em *R* (arquitetura RQRN1E), realizamos a migração completa para `PyTorch`, otimizando o fluxo de treinamento e adoção de recursos computacionais mais atuais. O processo inclui:

- **Ativações heterogêneas**. Emprega-se `GELU` nos blocos pré-τ (gradientes suaves) e `SiLU` nos blocos pós-τ, garantindo continuidade diferenciável em torno da injeção.
- **Regularização composta**. *Locked-dropout*, ruído gaussiano nas *features*, normalização espectral (λ_{sp}), *weight-decay* desacoplado e **Sharpness-Aware Minimization** (SAM) atuam em conjunto para mitigar o sobre-ajuste e achatar o vale da função de perda, isto é, buscar mínimos amplos de baixa curvatura, menos sensíveis a pequenas perturbações nos parâmetros.

Arquiteturas Alternativas (RQRN2E, RGRN):

Além da **RQRN1E**, implementamos e analisamos a **RQRN2E**, que usa duas redes em série para modularizar o papel de τ. Em paralelo, incluimos a **RGRN** (Rede Gaussiana para Regressão Quantílica), em que a rede infere μ e σ para aproximar qualquer quantil pela Normal ajustada. Tais arquiteturas foram implementadas a fim de comparar com os resultados do modelo RQRN1E.

3.1 Descrição dos Conjuntos de Dados

Foram empregados os mesmos *datasets* examinados por Zeco (2024), garantindo comparabilidade direta com os resultados anteriores. Abaixo descrevem-se as variáveis de cada conjunto e, em seguida, apresenta-se uma leitura concisa da distribuição da variável-alvo, ilustrada por gráficos de violino[2].

[2]Os arquivos PNG encontram-se em `imagens/`. As figuras são citadas no texto como Fig. 5–7.

(a) **Diamonds** Retirado do pacote `ggplot2` (*R*), contém 53 940 amostras de diamantes, cada uma com:

- *carat* (peso em quilates),
- *cut* (qualidade do corte, categórico),
- *color* (cor, categórico),
- *clarity* (clareza, categórico),
- *price* (US$) variável-resposta.

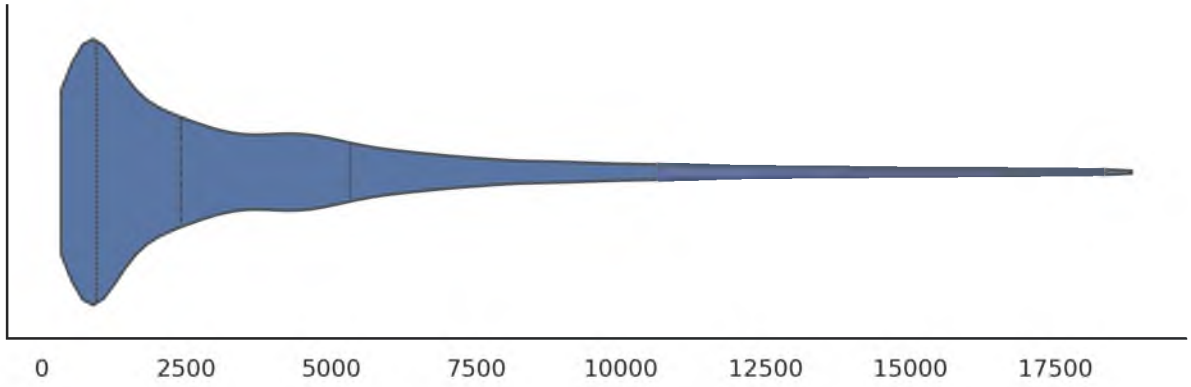

Figura 5: Distribuição da variável *price* (US$) no conjunto *Diamonds*.

A Fig. 5 revela assimetria pronunciada à direita: cerca de 75 % dos preços situam-se abaixo de US$ 5,3 mil, mas outliers raros elevam a cauda até quase US$ 19 mil.

(b) **Boston Housing** Disponível no pacote `MASS`, reúne 506 zonas residenciais de Boston. As covariáveis refletem condições socioeconômicas; o alvo é o valor mediano das habitações (*MEDV*, em milhares de US$).

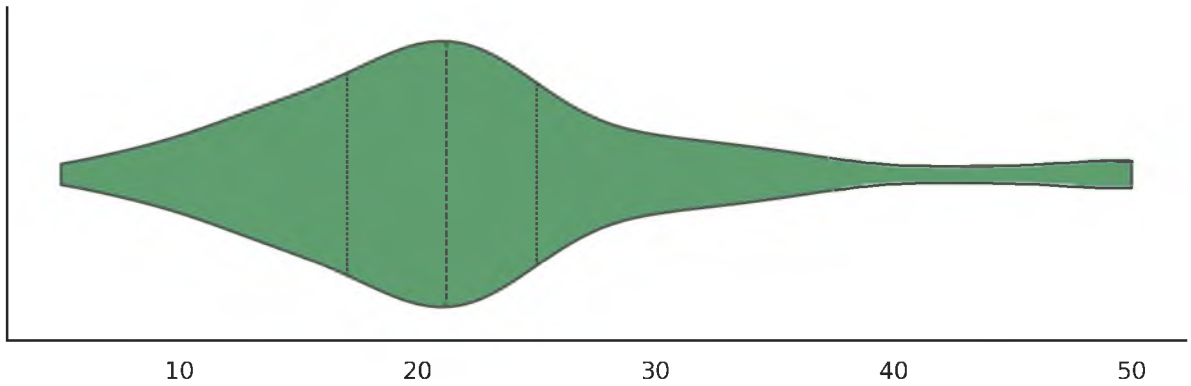

Figura 6: Distribuição da variável *MEDV* (US$ mil) no *Boston Housing*.

Como visto na Fig. 6, a distribuição é quase simétrica em torno de US$ 22 mil, com leve cauda à direita. O limite superior fixo em 50 evidencia-se como um corte

abrupto; fora isso, a variabilidade é moderada, justificando o uso de modelos lineares clássicos.

(c) **Yacht Hydrodynamics** Contém 308 experimentos hidrodinâmicos de iates, com variáveis geométricas (*longitudinal position*, *prismatic coefficient*, razões de comprimento/largura, etc.) e resistência residuária (N) como resposta.

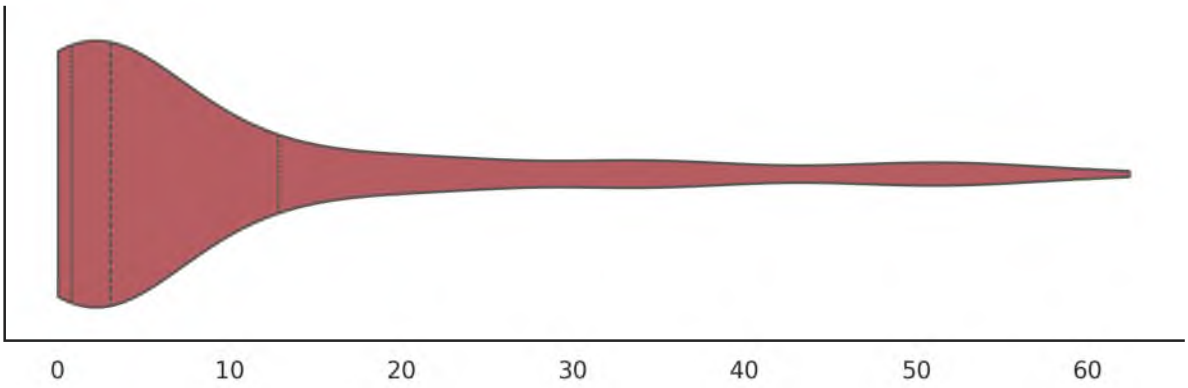

Figura 7: Distribuição da resistência residuária (N) no *Yacht Hydrodynamics*.

Na Fig. 7 observa-se forte concentração abaixo de 5 N; poucos ensaios, contudo, estendem a cauda até 62 N, fazendo a média (10,5 N) superar amplamente a mediana (3,1 N).

3.2 Pré-processamento e divisão dos dados

A preparação dos dados obedeceu a um protocolo em cinco etapas, sintetizado a seguir.

3.2.1 Filtragem e limpeza

Seguindo o trabalho de Zeco (2024), suprimiram-se:

(i) observações com valores ausentes;

(ii) registros com zeros fisicamente impossíveis (p. ex. peso do diamante igual a zero);

(iii) observações extremas que distorcessem a distribuição.

3.2.2 Codificação de variáveis qualitativas

Seguindo o trabalho de Zeco (2024), colunas nominais foram transformadas em indicadores binários $e_k(C) = 1\{C = k\}$, preservando a ausência de ordem natural entre

categorias e permitindo que cada modalidade contribua de forma independente para o modelo.

3.2.3 Padronização de covariáveis contínuas

Para cada atributo numérico x_j no subconjunto de treino $\mathcal{D}_{\text{tr}}$, foram estimados a média $\hat{\mu}_j$ e desvio-padrão $\hat{\sigma}_j$. A padronização foi definida como

$$z_j = \frac{x_j - \hat{\mu}_j}{\hat{\sigma}_j},$$

utilizando os mesmos $(\hat{\mu}_j, \hat{\sigma}_j)$ em validação e teste, o que evita vazamento de informação e assegura escalas homogêneas (HASTIEC et al., 2009).

3.2.4 Particionamento estratificado

Cada conjunto foi repartido em 70% treino, 20% validação e 10% teste. A amostragem em cada estrato baseou-se em decis da variável-alvo Y, isto é, a observação i pertence ao estrato

$$s_i = \sum_{q=1}^{10} 1\{y_i \leq F_Y^{-1}(q/10)\},$$

garantindo representatividade de todas as regiões da distribuição (KOENKER; BASSETT, 1978) . O conjunto de teste permaneceu intocado até a avaliação final.

3.2.5 Validação interna em blocos.

Dentro de $\mathcal{D}_{\text{tr}}$ empregou-se a validação cruzada em cinco blocos formados pela variável com maior coeficiente de variação, reduzindo dependência serial entre partições e, portanto, o viés otimista na estimação (PEDREGOSA et al., 2011).

3.3 Técnicas de regularização

A estratégia de controle de complexidade foi dividida em dois grupos: *(i)* regularizações **comuns**, aplicadas a *todas* as arquiteturas avaliadas, e *(ii)* regularizações **adicionais**, necessárias apenas para a RQRN1E.

Regularizações comuns

- **Early stopping** — mantido da tese de Zeco (2024) por sua eficácia em conjuntos pequenos (PRECHELT, 1998).
- **Decaimento dos pesos** (ℓ_2) — forma desacoplada de AdamW (LOSHCHILOV; HUTTER, 2019).
- **Clipping adaptativo do gradiente** — prevenção de explosões numéricas conforme (BROCK et al., 2021).

Regularizações adicionais da RQRN1E

As camadas monotônicas pós-τ tornaram a RQRN1E propensa a sobre-ajuste, levando à inclusão de quatro técnicas *não presentes* em (ZECO, 2024):

- **LockedDropout** — máscara fixa por passo temporal Merity et al. (2017).
- **Sharpness-Aware Minimization (SAM)** — busca regiões planas da perda Foret et al. (2021a).
- **Ruído de atributos** — regularização de Tikhonov sobre $\nabla_x f_\theta$ Bishop (1995).
- **Penalização espectral** — teto explícito à constante de Lipschitz Yoshida e Miyato (2017), Bartlett et al. (2017).

O trio *LockedDropout–SAM–ruído de atributos* foi decisivo para mitigar o sobre-ajuste oriundo das camadas monotônicas, ao passo que a penalização espectral estabilizou a busca bayesiana, resultando em menores perdas Pinball.

3.4 Inicialização dos pesos

A escolha do esquema de inicialização influencia diretamente a propagação de sinais, o condicionamento de Hessianas locais e, de modo agregado, a estabilidade do gradiente durante as primeiras iterações de treinamento. Neste trabalho adotaram-se políticas distintas, adequadas à natureza de cada arquitetura.

A consistência estatística dos sinais propagados nas primeiras iterações de treinamento foi assegurada por esquemas de amostragem que (i) preservam variância ao longo da profundidade e (ii) respeitam eventuais restrições de Lipschitz/monotonicidade. Seja d_{in} a dimensão de entrada de uma camada e d_{out} a de saída.

1. Arquitetura RQRN1E

Camadas anteriores a inserção do parâmetro τ

Cada transformação linear desses blocos é inicializada por uma distribuição uniforme

$$W_{ij} \sim \mathcal{U}\Big[-\tfrac{1}{\sqrt{d_{\text{in}}}}, \tfrac{1}{\sqrt{d_{\text{in}}}}\Big],$$

regra conhecida como *Xavier uniforme* Glorot e Bengio (2010a). Tal escolha garante que os sinais nem cresçam nem diminuam demais ao passar pelas camadas, melhorando a convergência do treinamento. É uma escolha padrão em muitas bibliotecas (como *Tensorflow* e *Pytorch*) para redes densas com funções de ativação simétricas.

Inicialização *Lipschitz–Glorot.*

Combina a distribuição uniforme de Glorot e Bengio (2010b) com a normalização espectral proposta por Anil et al. (2019a). Uma única passagem do *power method* seguida da projeção espectral de Miyato et al. (2018) e do re-escalonamento iterativo de Yoshida e Miyato (2017) mantém o operador dentro da cota 1-Lipschitz, condição necessária às garantias de generalização dependentes da norma espectral Bartlett et al. (2017). Os termos de viés são fixados em zero para não violar tal cota. Para detalhes técnicos completos, ver Anil et al. (2019a).

2. Arquitetura RQRN2E

A configuração de pesos adota as três estratégias descritas em Zeco (2024): *Xavier Uniform*, *He Kaiming* e *Orthogonal*. A variante *Xavier Normal* não foi considerada, pois apresenta variância idêntica à primeira.

- ***Xavier Uniform*** aplicada às camadas lineares do primeiro estágio quando se utilizam ativações simétricas (tanh ou *Swish*). São amostrados pesos segundo

  $$W_{ij} \sim \mathcal{U}\big[-\sqrt{6/(n_{\text{in}}+n_{\text{out}})},\ \sqrt{6/(n_{\text{in}}+n_{\text{out}})}\big],$$

  preservando a variância dos sinais ao longo da profundidade da rede (GLOROT; BENGIO, 2010a).

- ***He Kaiming*** empregada no segundo estágio após ativação por ReLU, com distribuição

  $$W_{ij} \sim \mathcal{U}\big[-\sqrt{6/n_{\text{in}}},\ \sqrt{6/n_{\text{in}}}\big] \quad \text{ou} \quad W_{ij} \sim \mathcal{N}\big(0, 2/n_{\text{in}}\big),$$

garantindo propagação estável de gradientes retificados (HE et al., 2015).

- ***Orthogonal*** reservada aos blocos de projeção 1×1 e à cabeça de regressão final; os pesos são obtidos por decomposição QR, o que preserva a ortogonalidade entre vetores de ativação e melhora a estabilidade numérica em arquiteturas profundas (SAXE et al., 2014).

3. Arquitetura RGRN

Na rede gaussiana (número de camadas $L \leq 4$) utilizou-se igualmente a amostragem *Xavier Uniform*. Esse esquema fixa a variância dos pesos em $\mathrm{Var}[W_{ij}] = 2/(n_{\text{in}}+n_{\text{out}})$, de modo que $\mathrm{Var}[z_{\text{out}}] \approx \mathrm{Var}[z_{\text{in}}]$ para cada camada, onde z denota o vetor de pré-ativações. Manter a mesma ordem de grandeza para as distribuições de entrada e saída — o que se denomina *balanceamento estatístico* — evita tanto o explodir quanto o esvanecer dos gradientes em arquiteturas rasas, dispensando restrições espectrais adicionais (GLOROT; BENGIO, 2010a).

3.5 Funções de ativação

Para conciliar regularidade teórica, estabilidade de treino e custo, adotou-se uma seleção dirigida de funções de ativação. Nos blocos *pré-*τ da RQRN1E usa-se a **GELU**, cuja suavidade amortiza o gradiente negativo mantendo constante de Lipschitz moderada; a própria injeção de τ passa antes pela **Probit**, padronizando o nível quantílico em escala gaussiana e evitando efeitos de τ*-dilution*; já nos blocos *pós-*τ aplica-se a **SiLU**, monótona e 1-Lipschitz, o que preserva a ordem dos quantis enquanto coopera com o *Sharpness-Aware Minimization*. Para as arquiteturas RQRN2E e RGRN, manteve-se a **ReLU**. A Tabela 1 resume as escolhas.

Tabela 1: Funções de ativação adotadas por arquitetura

Arquitetura	Localização da não-linearidade	Função $\varphi(\cdot)$
RQRN1E	Blocos pré-τ	GELU
	Injeção de τ	Probit
	Blocos pós-τ	SiLU
RQRN2E	Todos os blocos	ReLU
RGRN	Todos os blocos	ReLU

3.6 Otimizadores e agendadores de taxa de aprendizado

Para assegurar convergência estável e boa capacidade de generalização, adotou-se a seguinte política dual: (i) um otimizador de referência aplicado a *todas* as arquiteturas e (ii) uma variante refinada empregada exclusivamente na RQRN1E, em consonância com as regularizações adicionais já discutidas na Seção 3.3.

1. Otimizadores

Em todas as arquiteturas empregamos o *AdamW* (LOSHCHILOV; HUTTER, 2019), que combina taxas de aprendizagem adaptativas com *weight decay* desacoplado. Para a RQRN1E adicionamos o critério de *Sharpness-Aware Minimization* (SAM), aplicado sobre a própria atualização AdamW (FORET et al., 2021b). Essa combinação mostrou-se mais robusta a variações de gradiente e manteve regularização consistente entre os modelos avaliados. Detalhes completos sobre cada método podem ser encontrados nos trabalhos originais citados.

2. Agendador de taxa de aprendizado

Aquecimento linear e reinícios cossenoidais.

Um agendador de taxa de aprendizado (learning rate scheduler) é uma técnica usada durante o treinamento de redes neurais para ajustar dinamicamente a taxa de aprendizado ao longo das épocas. A ideia é melhorar a convergência e evitar problemas como overshooting ou estagnação.

Em geral, os agendadores exploram taxas de aprendizado altas no início do treinamento, a fim de explorar rapidamente o espaço de soluções. Posteriormente, empregam taxas menores no final para ajudar a refinar os pesos e alcançar um mínimo mais estável. Um agendador automatiza esse processo de forma inteligente.

Nesse trabalho, foi utilizado um aquecimento linear com reinícios cossenoidais. Mais especificamente, seja T o número total de *steps* e $\tau = 0.1T$ a duração da fase de aquecimento. A taxa instantânea segue

$$\eta_t = \begin{cases} \eta_{\max} \dfrac{t}{\tau}, & 0 \le t \le \tau, \\ \dfrac{\eta_{\max}}{2}\Big[1 + \cos\Big(\dfrac{\pi\,(t-\tau) \bmod T_i}{T_i}\Big)\Big], & t > \tau, \end{cases}$$

onde $T_0 = \lceil 1.2\, t_{\text{best}} \rceil$ é o primeiro comprimento de ciclo (t_{best} denota a época de menor

perda de validação obtida em experimento piloto) e T_i evolui segundo a política de *warm restarts* de Loshchilov et al. (2016). O aquecimento linear estabiliza as estimativas de momento nos primeiros passos, enquanto os reinícios cossenoidais exploram múltiplas regiões do espaço de parâmetros sem necessidade de redefinir hiper-parâmetros de forma manual.

Com essa combinação, garantiu-se convergência eficiente em todos os modelos, ao mesmo tempo em que se tirou proveito das peculiaridades dinâmicas da arquitetura recorrente.

3.7 Otimização de hiper-parâmetros

A escolha de hiper-parâmetros adequados é reconhecida como um dos fatores críticos para o desempenho de modelos de *deep learning* e, simultaneamente, um dos gargalos computacionais mais onerosos (JONES et al., 1998; SNOEK; LAROCHELLE; ADAMS, 2012). Entre as abordagens disponíveis, a Otimização Bayesiana (OB) distingue-se por seu caráter *sample-efficient*: ao iterar entre modelagem probabilística do espaço de busca e políticas de aquisição que equilibram exploração e obtém convergência exponencialmente mais rápida do que buscas ingênuas — atributo fundamental quando cada avaliação exige horas de treino em GPU (SHAHRIARI et al., 2016; FRAZIER, 2018).

Neste trabalho a OB é estruturada com as técnicas descritas a seguir:

1. *Desenho quase-aleatório inicial* — geração de amostras de baixa discrepância via sequência de Sobol, garantindo cobertura uniforme de Θ;
2. *Alocação adaptativa de recursos* — aplicação do *Successive Halving* para descartar precocemente configurações sub-ótimas;
3. *Atualização de crenças e decisão* — ciclo bayesiano que aproxima a métrica *Probability of Improvement* sem modelagem GP explícita;
4. *Espaços condicionais* — ativação seletiva de subespaços conforme a arquitetura candidata, reduzindo a dimensionalidade efetiva.

Cada sub-seção detalha o funcionamento de seu respectivo bloco, bem como a integração prática no `PyTorch/Optuna`. Juntas, essas camadas formam uma rotina de otimização que combina amostragem quase-uniforme, multi-fidelidade e condicionalidade.

3.7.1 Desenho quase–aleatório inicial

A otimização bayesiana precisa de pontos iniciais para começar a construir seu modelo probabilístico (geralmente um processo gaussiano). Em vez de escolher esses pontos aleatoriamente, é comum usar um desenho quase aleatório.

Para explorar o espaço de busca dos hiper-parâmetros de forma uniforme e eficiente, especialmente em alta dimensão, usar amostragem aleatória simples (como Monte Carlo) pode ser ineficiente. Uma alternativa é aplicar as sequências de baixa discrepância, como a sequência de Sobol (SOBOL', 1967).

Sobol é uma sequência determinística que gera pontos no espaço de forma quase uniforme, cobrindo o espaço de busca de maneira mais homogênea do que amostras aleatórias.

A sequência de Sobol é gerada diretamente no `PyTorch` por meio da classe `torch.quasirandom.SobolEngine`, que devolve pontos quase–aleatórios em Θ já no formato de tensor, prontos para serem passados para a *pipeline* de treinamento (PASZKE et al., 2019).

3.7.2 Alocação adaptativa de recursos

A alocação adaptativa de recursos é uma técnica eficiente para selecionar os melhores modelos ou configurações de hiper-parâmetros em um espaço de busca, economizando recursos computacionais.

Em vez de treinar todos os modelos até o fim (como em uma busca exaustiva), o procedimento *Successive Halving* – formulado na literatura de *bandits* por Jamieson et al. (2016) – e a sua extensão *Hyperband* Li et al. (2017), por exemplo, começa com muitos candidatos (opções de hiper-parâmetros), cada um com poucos recursos (utilização de poucas épocas de treino). O desempenho de todos é avaliado e descarta-se os piores (nesse caso, metade), redirecionando os recursos para os melhores candidatos. O processo é repetido até sobrar o melhor modelo.

Sendo assim, cada configuração θ é avaliada em fidelidades crescentes $r_0 < r_1 < \cdots < r_S$ (número de épocas de treino). No nível s calcula-se

$$\widehat{\mathcal{L}}_s(\theta) = \frac{1}{|\mathcal{B}_s|} \sum_{(x,y)\in\mathcal{B}_s} L_{\text{val}}^{(r_s)}(\theta\,;(x,y)),$$

e mantém-se apenas a fração $1/\eta$ das configurações com melhor desempenho, onde $\eta > 1$ é o fator de redução.

A implementação prática é feita com o auxilio do pacote *Optuna* (classe `SuccessiveHalvingPruner`), acoplada ao *loop* de treino do PyTorch via chamadas periódicas a `trial.report()` e `trial.should_prune()`.

3.7.3 Atualização de crenças e política de decisão

Embora não se recorra a uma modelagem de processo gaussiano explícito, o par *"Sobol + halving adaptativo"* mantém o ciclo bayesiano clássico:

(i) **Proposição**: extrai-se θ_t da sequência quase-aleatória;

(ii) **Observação**: obtém-se $\widehat{\mathcal{L}}_s(\theta_t)$ para múltiplos r_s;

(iii) **Revisão**: estima-se $\mathbb{P}\big[\mathcal{L}(\theta_t) < \mathcal{L}(\theta^{\star}_{\text{best}})\big]$;

(iv) **Decisão**: aloca-se recurso r_{s+1} ou podam-se as trajetórias dominadas.

Este mecanismo aproxima, de forma não paramétrica, a maximização de uma função de aquisição $\alpha_t(\theta) = \mathbb{P}\big[\mathcal{L}(\theta) < \mathcal{L}(\theta^{\star}_{\text{best}})\big]$, análoga ao *Probability of Improvement* da otimização bayesiana tradicional Shahriari et al. (2016).

3.7.4 Espaços condicionais

Para cada modelo, os hiper-parâmetros são amostrados da seguinte forma:

Dentro de Θ_j define-se o vetor $\theta_j = (\vartheta_1, \ldots, \vartheta_{d_j})$ cujas componentes são amostradas de forma independente segundo

$$\vartheta_k \sim \begin{cases} \mathit{Unif}(a_k,\, b_k), & \text{se } \vartheta_k \text{ for contínuo em escala linear,} \\ \mathit{LogUnif}(a_k,\, b_k), & \text{se } \vartheta_k \text{ variar em múltiplas ordens de grandeza,} \\ \mathit{Categorical}(\mathcal{C}_k), & \text{se } \vartheta_k \text{ pertencer a um domínio discreto finito.} \end{cases}$$

Esse esquema de *spaces by conditional activation* Bergstra et al. (2012), Falkner, Klein e Hutter (2018) garante que parâmetros irrelevantes permaneçam inativos, evitando avaliações inválidas e reduzindo a *dimensionalidade intrínseca* da busca. Além disso, a independência marginal das distribuições base facilita a incorporação de técnicas de amostragem sob restrições de orçamento, como *BOHB*, empregada neste trabalho.

3.7.5 Espaço de busca dos hiper-parâmetros

Antes de analisar o desempenho empírico, convém explicitar o domínio de otimização sobre o qual o processo bayesiano explorou configurações. A Tabela 2 sintetiza, para cada arquitetura, o tipo de cada hiperparâmetro, o intervalo (ou conjunto) considerado e indica, quando pertinente, se a amostragem ocorreu em escala logarítmica. Tal delineamento cumpre duas funções centrais:

(*i*) revelar as suposições a priori que informam o modelo de busca Frazier (2018)

(*ii*) permitir a reprodutibilidade de futuros estudos comparativos Bergstra et al. (2011), Bardenet et al. (2013).

Tabela 2: Espaço de busca bayesiana adotado para cada arquitetura

Arquitetura	**Parâmetro**	**Tipo**	**Faixa / Opções**	**log**	**Observações**
RQRN1E	L	int	$[3, 6]$	–	profundidade total
	h_0	int	$[64, 256]$	–	largura das camadas anteriores a inserção do τ
	h_{end}	float	$[0.25, 1.00]$	–	fração de h_0 nas camadas após inserção do τ
	τ_{layer}	cat	$\{1, \ldots, L-1\}$	–	posição de injeção de τ
	batch_size	cat	$\{32, 64, 128, 256\}$	–	tamanho do *batch*
	learning_rate	float	$[5 \times 10^{-5}, 5 \times 10^{-4}]$	✓	inicial
	weight_decay	float	$[1 \times 10^{-6}, 5 \times 10^{-2}]$	✓	(LOSHCHILOV; HUTTER, 2019)
	dropout_pre	float	$[0.0, 0.15]$	–	(FORET et al., 2021b)
	input_noise	float	$[0.0, 0.02]$	–	ruído (TAGASOVSKA et al., 2019)
	λ_{sp}	float	$[5 \times 10^{-5}, 1 \times 10^{-3}]$	✓	Parseval (CISSé et al., 2017)
RQRN2E	L_{base}	int	$[1, 4]$	–	estágio 1 (quantil)
	L_{recal}	int	$[1, 4]$	–	estágio 2 (recalibração)
	h_0	int	$[64, 256]$	–	largura inicial
	batch_size	cat	$\{32, 64, 128, 256\}$	–	tamanho do batch
	learning_rate	float	$[5 \times 10^{-5}, 10^{-3}]$	✓	BayesOpt (SNOEK et al., 2012; AKIBA et al., 2019)
RGRN	L	int	$[2, 5]$	–	profundidade
	h_0	int	$[64, 256]$	–	largura constante
	batch_size	cat	$\{32, 64, 128, 256\}$	–	tamanho do batch
	learning_rate	float	$[5 \times 10^{-5}, 10^{-3}]$	✓	taxa de aprendizado

3.7.6 Herança de arquitetura

Após a rodada inicial de otimização bayesiana do modelo de referência, verificou-se que o número de camadas permaneceu constante e que as larguras iniciais só variaram marginalmente entre os cinco *folds*. Fixou-se, portanto, a largura correspondente ao *fold* com menor perda de validação e essa configuração estrutural foi aplicada a todas as demais variantes. Desse modo, cada rede passa a diferir apenas pelo ponto de injeção do nível quantílico τ, de forma que qualquer diferença de desempenho discutida nos resultados reflete exclusivamente o efeito da posição de τ e não mudanças de capacidade ou profundidade do modelo.

No *Pytorch*, a herança é implementada copiando o dicionário de configuração `base_cfg` e fazendo *override* apenas nos elementos que permanecem livres, a função `suggest_*` condiciona automaticamente os novos valores à arquitetura ativa, reproduzindo o mecanismo formal descrito acima.

3.8 Validação cruzada aninhada e seleção do modelo final

Cada base foi submetida a uma *nested*-CV em duas camadas. A **camada externa** particiona todo o conjunto em K_{outer} blocos; em cada iteração o bloco $\mathcal{D}_{\text{test}}^{(k)}$ (10 % dos dados quando $K_{\text{outer}} = 1$, ou $1/K_{\text{outer}}$ do total, quando $K_{\text{outer}} > 1$) fica intocado para avaliação final, enquanto o complemento $\mathcal{D}_{\text{train}}^{(k)}$ alimenta a **camada interna**. Nessa camada a busca bayesiana (*Sobol + Successive-Halving*) executa uma k_{inner}-fold CV sobre os 70 % restantes, gerando a partição de validação (20 %) em cada fold interno. Após k_{inner} sugestões obtém-se a configuração $\theta_k^\star$ de menor perda média interna; com esses hiper-parâmetros treina-se novamente na íntegra de $\mathcal{D}_{\text{train}}^{(k)}$, aplicando todas as técnicas de regularização, resultando no preditor $f_k(x)$.

O processo externo gera, portanto, o conjunto de modelos independentes $\{f_k\}_{k=1}^{K_{\text{outer}}}$. Para cada amostra de teste calcula-se a *mediana* das predições $\widetilde{f}(x) = \text{median}\{f_k(x)\}$, formando um **median ensemble**. O risco final utilizado em todas as tabelas é a média das perdas desse ensemble sobre os blocos externos:

$$\widehat{R}_{\text{nested}} = \frac{1}{K_{\text{outer}}} \sum_{k=1}^{K_{\text{outer}}} L\big(\widetilde{f}, \mathcal{D}_{\text{test}}^{(k)}\big),$$

o que fornece estimativas praticamente não viesadas do risco e reduz a variância em $\sigma^2/(K_{\text{outer}}R)$ (NADEAU; BENGIO, 2003). Os valores de $(K_{\text{outer}}, k_{\text{inner}}, R)$ foram ajustados ao porte de cada conjunto: $(1, 5, 1)$ para *Boston* e *Yacht*, e $(3, 5, 3)$ para *Diamonds*, correspondendo a aproximadamente 500, 500 e 1500 treinos por arquitetura, respectiva-

mente, sem comprometer a tratabilidade computacional.

3.9 Modelos avaliados

Para quantificar o impacto da posição de injeção do parâmetro τ e contrastar arquiteturas de um e dois estágios, seis modelos (**M1**–**M6**) foram definidos.

M1 **RQRN1E$_{\text{opt}}$** (baseline otimizado). *Bayesian optimisation* sobre *todos* os hiper-parâmetros, inclusive a posição de τ. O procedimento devolve $L^\dagger$ camadas e larguras $(h_0^\dagger, h_{\text{end}}^\dagger)$, que servirão de referência para os modelos seguintes.

M2 **RQRN1E$_{\tau_{\text{input}}}$** (τ na entrada). Herda de M1 os valores fixos $\{L^\dagger, h_0^\dagger, h_{\text{end}}^\dagger\}$; τ é concatenado ao vetor de *features*. Todos os demais hiper-parâmetros (*learning rate*, dropout, ruído, *batch size*, etc.) são novamente otimizados pelo mesmo processo bayesiano.

M3 **RQRN1E$_{\tau_{\text{late}}}$** (τ na penúltima camada). Mantêm-se $L^\dagger$ e $(h_0^\dagger, h_{\text{end}}^\dagger)$ otimizado para o modelo M1; τ é injetado na penúltima camada. Demais hiper-parâmetros livres são otimizados de forma independente.

M4 **RQRN1E$_{\tau_{\text{last}}}$** (τ na última camada). Herda de M1 os valores fixos $\{L^\dagger, h_0^\dagger, h_{\text{end}}^\dagger\}$, com τ alimentando apenas a camada de saída. Todos os hiper-parâmetros *exceto* $(\{L^\dagger, h_0^\dagger, h_{\text{end}}^\dagger\})$ passam por otimização bayesiana.

M5 **RQRN2E** (rede em dois estágios). Ambos os estágios têm seus próprios conjuntos de hiper-parâmetros (larguras relativas, *learning rates*, etc.) otimizados conjuntamente.

M6 **RGRN** (baseline gaussiano). Modela média e desvio-padrão sob erro homocedástico. Todo o espaço de hiper-parâmetros pertinente (profundidade, largura, regularização, *learning rate*, ...) é explorado via otimização bayesiana.

Esta taxonomia permite comparar, de forma controlada, (i) o efeito da posição de τ em redes de estágio único, (ii) a vantagem (ou não) de condensar a estrutura em um único estágio, e (iii) o ganho sobre um modelo paramétrico gaussiano simples.

4 Resultados

Este capítulo consolida a avaliação empírica das três famílias de modelos (RQRN1E, RQRN2E e RGRN) nos *benchmarks Yacht*, *Boston* e *Diamonds*. Para cada *dataset* são apresentados:

(i) as arquiteturas dos cinco *folds* selecionadas pela Otimização Bayesiana;

(ii) o esforço computacional associado a cada modelo (épocas de *fine-tune* e tempo de treinamento).

(iii) as métricas de *treino*, *validação* e *teste*, reportadas como mediana±desvio-padrão nos cinco *folds* externos;

As seções seguintes apresentam, respectivamente, os resultados para *Yacht* (§4.1), *Boston* (§4.2) e *Diamonds* (§4.3), seguindo a estrutura uniforme: tabela de arquiteturas, tabelas de métricas (treino/validação e teste) e uma breve discussão comparativa dos desempenhos observados.

4.1 Yacht

Configurações experimentais

Todos os experimentos foram conduzidos no ambiente Google Colab Pro, empregando uma GPU NVIDIA A100 de 40 GB. A implementação em PyTorch 2.1 utilizou tensores em `torch.bfloat16`, explorando instruções *Tensor Core* para acelerar as etapas de *forward* e *backward* sem perda mensurável de precisão numérica (WANG et al., 2021).

Antes de apresentar as tabelas com as arquiteturas resultantes, sumarizamos as definições comuns a *todos* os bancos (*Diamonds*, *Boston* e *Yacht*).

- **Validação cruzada aninhada.** Cada banco é avaliado por NESTED-CV$(K_{\text{outer}}, k_{\text{inner}}) = (R, 5)$, repetida $R = 3$ vezes para o Diamonds e $R = 1$ vezes para os demais, como descrito na Seção 2.6.
- **Otimização Bayesiana.** Todos os hiper-parâmetros listados no Quadro 2 (incluindo η, p_{drop}, σ_{noise} e λ_{sp}) são ajustados por busca Bayesiana com *Sobol* + *Successive-Halving* (Seção 2.6).
- **Fine-tune.** Após fixar a arquitetura de cada *fold*, executa-se um ajuste final de ε épocas; as colunas *Opt.* e *Train* relatam, respectivamente, o tempo gasto na busca de hiper-parâmetros e no treino completo.

Convenções

Nas tabelas de arquitetura utilizam-se as abreviações:

L	número total de camadas;
τ	camada de inserção de τ;
p_{drop}	*locked-dropout*;
σ_{noise}	ruído aditivo nas *features*;
η	*learning rate*;
BS	*batch size*;
ε	épocas do ajuste final;
*	Identifica o exato local de inserção do τ;
Tempo (s)	tempo total (em segundos) gasto para efetuar a otimização e o treinamento no *fold*;

Na coluna "**Arquitetura**" emprega-se a forma geral

$$a\text{–}b : c_1 \longrightarrow d{:}* c_2 \longrightarrow \ldots,$$

em que

- $a\text{–}b$ indica o intervalo de camadas a até b (inclusive);
- c_k é o número de neurônios de cada camada do bloco;
- o asterisco "$*$" marca a camada onde τ é inserido;
- a seta "$\longrightarrow$" separa blocos com larguras distintas.

Exemplo

A cadeia

$$\boxed{1\text{–}3 : 128 \longrightarrow 4{:}* 25 \longrightarrow 5{:}25}$$

perfila uma rede de cinco camadas: as três primeiras contêm 128 neurônios cada; as duas últimas, 25 neurônios; e a concatenação de τ ocorre exatamente na quarta camada.

A Tabela 3 a seguir apresenta os detalhes das configurações dos hiper-parâmetros para cada modelo em cada um dos cinco *folds*.

Tabela 3: Arquiteturas geradas em cada um dos cinco *folds* para o banco *Yacht*

modelo	Fold	L	τ	Arquitetura (camada:neurônios)	Reg. p_{drop}	Reg. σ_{noise}	η	BS	ε	Tempo (s)
$\mathbf{RQRN1E}_{\tau_{\text{opt}}}$	Fold 1	5	4	1–3:144 → 4: ⋆24→5:24	.030	.014	.0029	64	23	15
	Fold 2			1–3:145 → 4: ⋆22→5:22	.031	.014	.0029	64	23	18
	Fold 3			1–3:146 → 4: ⋆23→5:23	.031	.014	.0029	64	20	21
	Fold 4			1–3:147 → 4: ⋆23→5:23	.030	.014	.0028	64	22	15
	Fold 5			1–3:146 → 4: ⋆23→5:23	.029	.014	.0028	64	21	15
$\mathbf{RQRN1E}_{\tau_{\text{input}}}$	Fold 1	5	1	1: ⋆144→2–3:144→4-5:24	0	.0021	.0033	64	19	8
	Fold 2				0	.0022	.0035	64	18	7
	Fold 3				0	.0024	.0035	64	18	8
	Fold 4				0	.0025	.0032	64	23	5
	Fold 5				0	.0027	.0038	64	21	10
$\mathbf{RQRN1E}_{\tau_{\text{late}}}$	Fold 1	5	4	1–3:144 → 4: ⋆24→5:24	.031	.009	.0022	64	15	13
	Fold 2				.032	.009	.0022	64	17	11
	Fold 3				.033	.010	.0021	64	24	9
	Fold 4				.035	.010	.0021	64	20	10
	Fold 5				.036	.011	.0020	64	18	15
$\mathbf{RQRN1E}_{\tau_{\text{last}}}$	Fold 1	5	5	1–3:144 → 4:24 → 5: ⋆24	.005	.014	.0036	64	18	13
	Fold 2				.006	.014	.0034	64	22	16
	Fold 3				.006	.015	.0037	64	21	13
	Fold 4				.007	.016	.0036	64	19	14
	Fold 5				.007	.013	.0035	64	19	10
RQRN2E	Fold 1	4	–	1:88 →2:33 → 3: ⋆24 → 4:24	–	–	.0021	128	72	105
	Fold 2			1:89 →2:32 → 3: ⋆24 → 4:24	–	–	.0020	128	82	95
	Fold 3			1:90 →2:32 → 3: ⋆24 → 4:24	–	–	.0020	128	86	112
	Fold 4			1:91 →2:31 → 3: ⋆24 → 4:24	–	–	.0020	128	77	110
	Fold 5			1:90 →2:32 → 3: ⋆24 → 4:24	–	–	.0020	128	71	104
RGRN	Fold 1	3	–	1–3:170	–	–	.0087	64	31	12
	Fold 2			1–3:172	–	–	.0086	64	33	14
	Fold 3			1–3:171	–	–	.0086	64	35	12
	Fold 4			1–3:173	–	–	.0085	64	32	15
	Fold 5			1–3:171	–	–	.0085	64	30	15

A Tabela 3 evidencia a convergência: o modelo $\text{RQRN1E}_{\text{OPT}}$ recupera, em todos os *folds*, a mesma topologia com $L = 5$ camadas, largura inicial $h_0 \approx 145$ neurônios e inserção de τ na penúltima camada, maximizando a capacidade de representação antes da projeção 1-Lipschitz final. A soma dos tempos de busca e treino varia apenas de 15 s a 21 s (média 17 s), sendo $\sim 82\,\%$ desse total dedicado à otimização bayesiana. Mesmo assim, o pior tempo observado na $\text{RQRN1E}_{\text{OPT}}$ é cerca de **25** % menor que o maior valor registado

para o modelo em dois estágios RQRN2E (112 s). Conforme explicado anteriormente, nota-se que a arquitetura do modelo $\text{RQRN1E}_{\text{OPT}}$ foi replicada para as demais variações RQRN1E a fim de avaliar a sensibilidade do modelo na posição de inserção do parâmetro τ. Todos os modelos operam com $p_{\text{drop}} \leq 0.036$ e $\sigma_{\text{noise}} \leq 0.014$, faixa que assegura estabilidade em `bfloat16`/*Tensor Cores* sem perda de precisão, conforme demonstrado por (WANG et al., 2021).

A Tabela 4 apresenta as métricas das arquiteturas supracitadas.

Tabela 4: Desempenho nas partições de treino e validação (*Yacht*)

modelo	**Etapa**	RMSE	MAE	Pinball	Cob.(%)	Calib.	Cruz.(%)
$\text{RQRN1E}_{\text{opt}}$	Treino	**.58 ± .14**	.57 ± .20	**.16 ± .10**	**93.2 ± .2**	**.006 ± .001**	0
	Validação	**.60 ± .18**	**.62 ± .21**	**.17 ± .11**	**94.1 ± .3**	**.005 ± .002**	0
$\text{RQRN1E}_{\tau_{\text{input}}}$	Treino	2.42 ± 1.05	1.68 ± .90	.95 ± .21	89.2 ± 3.7	.016 ± .012	0
	Validação	2.75 ± 1.62	1.89 ± .86	1.01 ± .24	86.5 ± 4.5	.017 ± .015	0
$\text{RQRN1E}_{\tau_{\text{late}}}$	Treino	.63 ± .18	.60 ± .21	.18 ± .10	92.5 ± .6	.007 ± .002	0
	Validação	.70 ± .22	.66 ± .22	.19 ± .11	93.2 ± .7	.007 ± .002	0
$\text{RQRN1E}_{\tau_{\text{last}}}$	Treino	.83 ± .36	.69 ± .20	.28 ± .11	93.3 ± 6.2	.013 ± .009	0
	Validação	.98 ± .43	.72 ± .38	.32 ± .18	92.6 ± 5.8	.015 ± .010	0
RQRN2E	Treino	.72 ± .41	.68 ± .20	.28 ± .14	92.5 ± 1.3	.031 ± .002	2.30 ± .05
	Validação	.79 ± .48	.71 ± .22	.32 ± .16	93.0 ± 1.5	.038 ± .003	3.05 ± .07
RGRN	Treino	.79 ± 1.44	**.41 ± .91**	.19 ± .18	94.4 ± 2.43	.027 ± .021	0
	Validação	1.22 ± .99	.93 ± .63	.23 ± .20	95.28 ± 2.76	.015 ± .023	0

A Tabela 4 mostra que o melhor desempenho fica com a $\text{RQRN1E}_{\text{OPT}}$: erros pontuais mínimos, *Pinball* $0.16 \rightarrow 0.17$, cobertura próximo meta e calibração muito baixa (< 0.01). A inserção do τ na penúltima camada, assim como no modelo $\text{RQRN1E}_{\text{OPT}}$, evidencia que a variante (τ_{LATE}) quase não altera o quadro de métricas. As variantes τ_{INPUT} e τ_{LAST} sofrem queda acentuada: Pinball $\times 5$ e cobertura $< 90\,\%$. A RQRN2E apresenta uma calibração próxima da RGRN mas ainda marginalmente distante dos modelos de um estágio, além de introduzir $3\,\%$ de cruzamentos.

Por conseguinte, vemos na Tabela 5 os resultados das métricas no conjunto de teste.

Tabela 5: Desempenho final no conjunto de teste (*Yacht*)

modelo	RMSE	MAE	Pinball	Cob.(%)	Calib.	Cruz.(%)
$\text{RQRN1E}_{\text{opt}}$	$\mathbf{.65 \pm .15}$	$\mathbf{.64 \pm .23}$	$\mathbf{.20 \pm .10}$	$\mathbf{95.0 \pm .2}$	$\mathbf{.008 \pm .010}$	0
$\text{RQRN1E}_{\tau_{\text{input}}}$	2.98 ± 1.01	$1.97 \pm .94$	$1.14 \pm .80$	82.6 ± 5.6	$.017 \pm .018$	0
$\text{RQRN1E}_{\tau_{\text{late}}}$	$.68 \pm .20$	$.72 \pm .21$	$.25 \pm .13$	$93.5 \pm .9$	$.009 \pm .011$	0
$\text{RQRN1E}_{\tau_{\text{last}}}$	$.99 \pm .33$	$.79 \pm .21$	$.34 \pm .08$	92.3 ± 3.7	$.017 \pm .010$	0
RQRN2E	$.80 \pm .48$	$.70 \pm .22$	$.26 \pm .12$	93.0 ± 1.7	$.040 \pm .005$	$3.55 \pm .08$
RGRN	$1.29 \pm .61$	$.75 \pm .36$	$.25 \pm .22$	95.8 ± 2.3	$.025 \pm .017$	0

Como resume a Tabela 5, a $\text{RQRN1E}_{\text{OPT}}$ continua melhor: menor RMSE (0.65), Pinball (0.20) e calibração (0.008) com cobertura ideal (95 %). A versão τ_{LATE} entrega novamente um desempenho muito próximo do modelo ótimo. Já τ_{LAST} e τ_{INPUT} perdem progressivamente a forma dos intervalos: Pinball salta para 0.34 e 1.14, enquanto a cobertura cai a 92 e 83 %, respectivamente. A RQRN2E mantém erros moderados, mas com uma calibração (Calib. $\approx$ 0.40) um pouco pior que os outros modelos , todavia , com menor variabilidade entre *folds* e 3.5 % de cruzamentos. Por fim, a RGRN retorna a segunda pior *Pinball* e uma cobertura novamente um pouco inflacionada, confirmando que a hipótese gaussiana não é adequada para o *Yacht*.

4.2 Boston

Podemos visualizar as arquiteturas ótimas encontradas no processo de otimização bayesiana, com o auxilio da Tabela 6 a seguir.

Tabela 6: Arquiteturas geradas em cada um dos cinco *folds* para o banco *Boston*

modelo	Fold	L	τ	Arquitetura (camada:neurônios)	Reg. p_{drop}	Reg. σ_{noise}	η	BS	ε	Tempo (s)
	Fold 1	3	2	1:71→2: *15→3:15	.091	.0023	.00162	64	19	31
	Fold 2			1:72→2: *14→3:14	.085	.0021	.00178	64	17	32
RQRN1E$_{\tau_{\text{opt}}}$	Fold 3			1:73→2: *15→3:15	.088	.0021	.00207	64	20	34
	Fold 4			1:72→2: *14→3:14	.087	.0012	.00197	64	18	35
	Fold 5			1:71→2: *15→3:15	.089	.0019	.00183	64	21	37
	Fold 1	3	1	1: *71→2-3:15	0	.0017	.00215	64	20	16
	Fold 2				0	.0019	.00204	64	22	19
RQRN1E$_{\tau_{\text{input}}}$	Fold 3				0	.0017	.00207	64	25	21
	Fold 4				0	.0018	.00210	64	27	22
	Fold 5				0	.0019	.00212	64	30	24
	Fold 1	3	2	1:71→2: *15→3:15	.048	.0030	.00182	64	22	13
	Fold 2				.050	.0030	.00180	64	25	19
RQRN1E$_{\tau_{\text{late}}}$	Fold 3				.051	.0032	.00180	64	28	22
	Fold 4				.053	.0032	.00178	64	23	17
	Fold 5				.054	.0033	.00178	64	22	19
	Fold 1	3	3	1:71→2:15→3: *15	.031	.0044	.00183	64	24	13
	Fold 2				.032	.0045	.00186	64	20	16
RQRN1E$_{\tau_{\text{last}}}$	Fold 3				.033	.0046	.00180	64	21	13
	Fold 4				.034	.0047	.00195	64	27	14
	Fold 5				.035	.0048	.00192	64	28	13
	Fold 1	4	–	1:130→2:40→3: *28→4:28	–	–	.00027	128	100	88
	Fold 2			1:129→2:41→3: *27→4:28	–	–	.00027	128	103	91
RQRN2E	Fold 3			1:131→2:39→3: *28→4:28	–	–	.00027	128	103	85
	Fold 4			1:130→2:40→3: *28→4:28	–	–	.00027	128	104	84
	Fold 5			1:132→2:42→3: *29→4:28	–	–	.00027	128	110	91
	Fold 1	5	–	1–5:143	–	–	.00610	128	33	13
	Fold 2			1–5:146	–	–	.00605	128	36	14
RGRN	Fold 3			1–5:144	–	–	.00600	128	38	14
	Fold 4			1–5:147	–	–	.00595	128	40	15
	Fold 5			1–5:145	–	–	.00600	128	40	15

A Tabela 6 indica que a busca Bayesiana, para o *Boston Housing*, quase sempre converge para uma rede rasa com $L = 3$ camadas, largura inicial $h_0 \approx 72$ e a inserção do τ no segundo bloco.

O tempo combinado de otimização e treino do modelo RQRN1E$_{\text{OPT}}$ varia de 31 s a 37 s por *fold* (média 34 s). A versão em dois estágios RQRN2E mantém-se como a mais dispendiosa (84–91 s), enquanto o modelo RGRN encerra cada ciclo em aproximadamente

13–15 s.

A Tabela 7 explicíta os resultados das métricas nos conjuntos de treino e validação.

Tabela 7: Desempenho nas partições de treino e validação (*Boston*)

modelo	**Etapa**	RMSE	MAE	Pinball	Cob.(%)	Calib.	Cruz.(%)
$\text{RQRN1E}_{\text{opt}}$	Treino	$\mathbf{1.60 \pm .18}$	$\mathbf{0.96 \pm .34}$	$\mathbf{0.25 \pm .08}$	94.8 ± 0.3	$\mathbf{.009 \pm .003}$	0
	Validação	$\mathbf{1.72 \pm .09}$	$\mathbf{1.00 \pm .36}$	$\mathbf{0.28 \pm .10}$	$\mathbf{94.3 \pm 0.6}$	$\mathbf{.011 \pm .002}$	0
$\text{RQRN1E}_{\tau_{\text{input}}}$	Treino	3.65 ± 1.8	2.84 ± 1.8	$1.36 \pm .86$	90.6 ± 2.3	$.018 \pm .006$	0
	Validação	$4.12 \pm .93$	3.09 ± 1.4	$1.86 \pm .79$	88.3 ± 2.8	$.021 \pm .007$	0
$\text{RQRN1E}_{\tau_{\text{late}}}$	Treino	$1.62 \pm .17$	$0.98 \pm .35$	$0.29 \pm .09$	94.0 ± 0.7	$.011 \pm .004$	0
	Validação	$1.75 \pm .12$	$1.04 \pm .38$	$0.32 \pm .11$	94.1 ± 1.0	$.012 \pm .006$	0
$\text{RQRN1E}_{\tau_{\text{last}}}$	Treino	$2.74 \pm .28$	$1.67 \pm .17$	$0.69 \pm .06$	93.8 ± 1.7	$.015 \pm .008$	0
	Validação	$2.87 \pm .34$	$1.88 \pm .21$	$0.73 \pm .11$	92.5 ± 2.2	$.016 \pm .009$	0
RQRN2E	Treino	$2.10 \pm .50$	$1.42 \pm .37$	$0.61 \pm .15$	92.2 ± 1.5	$.020 \pm .033$	$2.00 \pm .07$
	Validação	$2.25 \pm .55$	$1.57 \pm .40$	$0.66 \pm .17$	91.1 ± 2.1	$.023 \pm .048$	$2.18 \pm .08$
RGRN	Treino	$1.73 \pm .32$	$0.93 \pm .24$	$0.72 \pm .19$	95.9 ± 1.3	$.072 \pm .023$	0
	Validação	$1.87 \pm .35$	$0.97 \pm .27$	$0.78 \pm .24$	96.4 ± 1.4	$.094 \pm .025$	0

À luz da Tabela 7, a $\text{RQRN1E}_{\text{OPT}}$ segue como referência: combina os menores erros pontuais (RMSE $= 1.60 \pm 0.18$ em treino e 1.72 ± 0.09 em validação) à menor *Pinball Loss* (0.25 ± 0.08 e 0.28 ± 0.10), mantendo cobertura média $94 \pm 1\,\%$ e calibração praticamente ideal (≤ 0.011). A injeção tardia de τ (τ_{LATE}) mantém, como esperado, a proximidade nas métricas o que qualifica o modelo $\text{RQRN1E}_{\text{LATE}}$.

Quando τ é fornecido como entrada (τ_{INPUT}) ou restrito à camada final (τ_{LAST}), o gargalo estrutural reduz a capacidade de ajuste: a *Pinball* cresce de duas a três vezes, a cobertura cai para $88-93\,\%$ e o erro de calibração dobra.

O esquema em dois estágios (RQRN2E) agrava esses problemas: além de erros mais altos, exibe calibração similar à variante τ_{late} e $2\,\%$ de cruzamentos, indicando falhas na ordem estocástica.

Por fim, a RGRN atinge uma cobertura um pouco acima da cobertura alvo e calibração 8 vezes pior que o modelo $\text{RQRN1E}_{\text{OPT}}$, naturalmente a presunção de normalidade é falha para esse conjunto de dados e esse modelo mais simples é inadequado.

A Tabela 8 a seguir, apresenta os resultados das métricas no conjunto de teste.

Tabela 8: Desempenho final no conjunto de teste (*Boston*)

modelo	RMSE	MAE	Pinball	Cob.(%)	Calib.	Cruz.(%)
$\text{RQRN1E}_{\text{opt}}$	$\mathbf{1.70 \pm .22}$	$1.07 \pm .32$	$\mathbf{0.34 \pm .08}$	$\mathbf{94.6 \pm 0.5}$	$\mathbf{.012 \pm .002}$	0
$\text{RQRN1E}_{\tau_{\text{input}}}$	$4.32 \pm .83$	$3.25 \pm .88$	$1.92 \pm .85$	87.5 ± 3.1	$.022 \pm .009$	0
$\text{RQRN1E}_{\tau_{\text{late}}}$	$1.77 \pm .25$	$1.14 \pm .29$	$0.37 \pm .09$	94.5 ± 0.9	$.015 \pm .008$	0
$\text{RQRN1E}_{\tau_{\text{last}}}$	$2.89 \pm .27$	$1.95 \pm .38$	$0.83 \pm .12$	$92.3 \pm .8$	$.017 \pm .010$	0
RQRN2E	$2.17 \pm .53$	$1.50 \pm .38$	$0.64 \pm .16$	92.0 ± 2.3	$.024 \pm .012$	$3.12 \pm .12$
RGRN	$1.98 \pm .32$	$\mathbf{1.03 \pm .08}$	$.87 \pm .16$	$97.2 \pm .8$	$.097 \pm .022$	0

No teste (Tabela 8), a $\text{RQRN1E}_{\text{OPT}}$ concentra o melhor equilíbrio: menor RMSE (1.70), Pinball (0.34) e cobertura ajustada (94.6 %) com calibração de 0.012, sem cruzamentos. A variante τ_{late} mantém desempenho semelhante (RMSE +4 %, Pinball +9 %) a um custo computacional menor. Já τ_{last} e τ_{input} sofrem degradação progressiva: Pinball cresce 2–5 vezes, cobertura cai para 92–88 % e o erro de calibração dobra. O arranjo em dois estágios (RQRN2E) piora ainda mais, com calibração correspondente ao dobro da encontrada na variante τ_{opt} e 3 % de cruzamentos. Embora a RGRN mantenha MAE ligeiramente inferior e uma *Pinball* próxima da variante τ_{last}, ainda entrega cobertura fora da faixa alvo e uma calibração ruim.

4.3 Diamonds

O conjunto *Diamonds* contém mais de 5.4×10^4 observações, o que torna o ajuste exaustivo de hiper-parâmetros computacionalmente oneroso. Conforme recomendado por Feurer e Hutter (2019), a busca bayesiana foi executada em uma sub-amostra de $n = 10\,000$ instâncias obtidas por amostragem estratificada nos décis da variável-alvo (*price*). Tal estratégia preserva a distribuição conjunta $p(X, Y)$ e garante que a ordem de preferência entre configurações estimada na amostra converge, em probabilidade, para a ordem no conjunto completo (teorema 2 em Klein et al. (2017)), obtendo-se redução superior a 50% no tempo total de otimização, sem perda estatística apreciável.

A seguir, a Tabela 9 apresenta as arquiteturas encontradas com o auxilio do processo de otimização bayesiana que foi sucedido a essa prévia amostragem no conjunto *Diamonds*.

Tabela 9: Arquiteturas geradas em cada um dos cinco *folds* para o banco *Diamonds*

modelo	Fold	L	τ	Arquitetura (camada:neurônios)	Reg. p_{drop}	Reg. σ_{noise}	η	BS	ε	**Tempo (s)**
	Fold 1	4	3	1–2:118 → 3: ⋆22→4:22	.062	.032	.00190	256	27	513
	Fold 2			1–2:119 → 3: ⋆24→4:24	.061	.033	.00188	256	26	552
RQRN1E$_{\tau_{\text{opt}}}$	Fold 3			1–2:117 → 3: ⋆20→4:20	.063	.031	.00187	256	30	534
	Fold 4			1–2:118 → 3: ⋆21→4:21	.062	.033	.00186	256	32	538
	Fold 5			1–2:118 → 3: ⋆23→4:23	.062	.032	.00187	256	31	529
	Fold 1	4	1	1: ⋆118→2:118→ 3-4:22	0	.020	.00210	256	23	337
	Fold 2				0	.019	.00209	256	21	330
RQRN1E$_{\tau_{\text{input}}}$	Fold 3				0	.018	.00212	256	28	342
	Fold 4				0	.017	.00208	256	25	344
	Fold 5				0	.017	.00209	256	29	310
	Fold 1	4	3	1–2:118 → 3: ⋆22→4:22	.025	.045	.00150	256	30	386
	Fold 2				.025	.044	.00148	256	34	382
RQRN1E$_{\tau_{\text{late}}}$	Fold 3				.026	.044	.00147	256	38	376
	Fold 4				.027	.043	.00146	256	42	380
	Fold 5				.027	.043	.00146	256	46	392
	Fold 1	4	4	1–2:118 → 3:22→4: ⋆22	.045	.036	.00190	256	25	332
	Fold 2				.044	.035	.00194	256	29	330
RQRN1E$_{\tau_{\text{last}}}$	Fold 3				.043	.034	.00192	256	21	325
	Fold 4				.043	.033	.00198	256	31	323
	Fold 5				.042	.033	.00197	256	30	337
	Fold 1	5	–	1:167→ 2:180→3:187→4: ⋆20→5:20	–	–	.00198	128	158	1001
	Fold 2			1:168→ 2:187→3:184→4: ⋆23→5:20	–	–	.00196	128	155	1032
RQRN2E	Fold 3			1:165→ 2:193→3:194→4: ⋆20→5:20	–	–	.00195	128	162	1017
	Fold 4			1:169→ 2:181→3:186→4: ⋆21→5:21	–	–	.00194	128	160	1030
	Fold 5			1:167→ 2:201→3:203→4: ⋆27→5:20	–	–	.00193	128	153	1037
	Fold 1	4	–	1–4:100	–	–	.00105	64	90	536
	Fold 2			1–4:112	–	–	.00104	64	100	604
RGRN	Fold 3			1–4:109	–	–	.00103	64	105	544
	Fold 4			1–4:105	–	–	.00103	64	102	471
	Fold 5			1–4:102	–	–	.00103	64	100	518

Como indica a Tabela 9, a Otimização Bayesiana voltou a convergir, no *Diamonds*, para redes com quatro camadas ($L = 4$) e largura inicial próxima de $h_0 \approx 118$ neurônios, arquitetura que todos os modelos RQRN1E passaram a herdar inalterada. O tempo médio por *fold* da configuração ótima ($\sim 533\,\text{s} \approx 8.9$ min) permanece cerca de **48 %** abaixo do gasto pela versão em dois estágios RQRN2E ($\sim 1\,024\,\text{s}$). Realocar o nível quantílico para a *entrada* reduz esse custo em aproximadamente **35 %**, ao passo que transferi-lo para a camada final mantém despesas praticamente idênticas às do modelo

ótimo.

A seguir, a Tabela 10 apresenta os resultados detalhados das métricas nos conjuntos de treino e validação.

Tabela 10: Desempenho nas partições de treino e validação (*Diamonds*)

modelo	**Etapa**	RMSE	MAE	Pinball	Cob.(%)	Calib.	Cruz.(%)
$\text{RQRN1E}_{\text{opt}}$	Treino	$\mathbf{489 \pm 11}$	$\mathbf{191 \pm 7}$	$\mathbf{89.4 \pm 3.5}$	$\mathbf{95.2 \pm 1.0}$	$\mathbf{.010 \pm .004}$	0
	Validação	$\mathbf{492 \pm 14}$	$\mathbf{200 \pm 8}$	$\mathbf{92.9 \pm 3.7}$	$\mathbf{95.3 \pm 1.2}$	$\mathbf{.012 \pm .005}$	0
$\text{RQRN1E}_{\tau_{\text{input}}}$	Treino	542 ± 24	284 ± 18	121.7 ± 9.4	90.7 ± 7.5	$.023 \pm .009$	0
	Validação	556 ± 22	291 ± 21	128.0 ± 10.6	88.3 ± 9.7	$.025 \pm .008$	0
$\text{RQRN1E}_{\tau_{\text{late}}}$	Treino	491 ± 16	201 ± 7	91.1 ± 1.9	95.3 ± 1.5	$.011 \pm .004$	0
	Validação	495 ± 15	204 ± 6	95.5 ± 2.1	95.0 ± 1.3	$.013 \pm .006$	0
$\text{RQRN1E}_{\tau_{\text{last}}}$	Treino	504 ± 13	207 ± 5	93.8 ± 2.1	90.8 ± 1.2	$.017 \pm .005$	0
	Validação	510 ± 11	209 ± 6	96.3 ± 2.8	92.7 ± 0.8	$.018 \pm .007$	0
RQRN2E	Treino	562 ± 20	294 ± 12	127.0 ± 5.0	65.2 ± 4.0	$.018 \pm .014$	3.1 ± 0.1
	Validação	578 ± 22	303 ± 13	128.8 ± 5.3	63.0 ± 4.5	$.020 \pm .018$	4.3 ± 0.1
RGRN	Treino	528 ± 18	279 ± 11	124.6 ± 4.5	95.9 ± 3.5	$.059 \pm .011$	0
	Validação	542 ± 16	281 ± 9	126.9 ± 4.8	96.2 ± 2.8	$.064 \pm .013$	0

Os valores da Tabela 10 confirmam que a $\text{RQRN1E}_{\text{OPT}}$ continua oferecendo o melhor *trade–off* entre acurácia pontual e qualidade dos intervalos.O modelo apresenta os menores erros médios (RMSE = 489±11 e 492±14; MAE = 191±7 e 200±8) e a menor *Pinball Loss* (89.4±3.5 e 92.9±3.7) nas partições de treino e validação, respectivamente. Os intervalos preditivos permanecem estreitos, com cobertura 95.2±1.0 % e 95.3±1.2 %, bem próximos à meta de 95 %, além da melhor calibração (0.010±0.004 e 0.012±0.005) e ausência de cruzamentos.

Inserir o τ na entrada ($\text{RQRN1E}_{\tau_{\text{INPUT}}}$) mantendo a arquitetura encontrada pela $\text{RQRN1E}_{\text{OPT}}$ limita a largura efetiva da rede e piora sensivelmente o desempenho: a *Pinball* sobe cerca de +36 % no treino e +38 % na validação, enquanto a cobertura cai quase 7 p.p. e o erro de calibração dobra.

O arranjo $\text{RQRN1E}_{\tau_{\text{LATE}}}$, que injeta τ na mesma camada do modelo otimizado, degrada pouco: *Pinball* cresce apenas +2–3 % e a cobertura mantém-se em torno de 95 %;surgindo como alternativa caso se deseje evitar o *overhead* de otimização do modelo *opt*.

Quando τ é adicionado apenas na última camada (τ_{LAST}), os pesos finais mostram maior variabilidade, resultando em aumentos de +5 % na *Pinball*, perda de 2–3 p.p. de cobertura e pior calibração (0.018±0.007).

O esquema em dois estágios (RQRN2E) volta a exibir as piores estatísticas: erros de calibração acima de 0.13, cobertura inferior a 65 % e 3–4 % de cruzamentos — reflexo da tendência de sobre-ajuste aos quantis extremos reportada por Kuleshov et al. (2022b).

Por fim, a RGRN retoma com a pior calibração e uma variabilidade que reforça a dificuldade de atingir a cobertura alvo, reiterando a superioridade global da $\text{RQRN1E}_{\text{OPT}}$.

A Tabela 11 dispõe os resultados das métricas no conjunto de teste.

Tabela 11: Desempenho final no conjunto de teste (*Diamonds*)

modelo	RMSE	MAE	Pinball	Cob.(%)	Calib.	Cruz.(%)
$\text{RQRN1E}_{\text{opt}}$	$\mathbf{503.2 \pm 4.0}$	$\mathbf{204.1 \pm 2.5}$	$\mathbf{94.3 \pm 3.3}$	$\mathbf{95.1 \pm 1.4}$	$\mathbf{.014 \pm .005}$	0
$\text{RQRN1E}_{\tau_{\text{input}}}$	568.5 ± 15.8	299.8 ± 21.0	134.7 ± 8.9	88.7 ± 11.0	$.041 \pm .012$	0
$\text{RQRN1E}_{\tau_{\text{late}}}$	510.3 ± 5.0	205.4 ± 3.1	95.3 ± 3.7	95.0 ± 1.8	$.021 \pm .006$	0
$\text{RQRN1E}_{\tau_{\text{last}}}$	517.3 ± 9.0	214.6 ± 5.2	97.2 ± 3.8	93.5 ± 2.7	$.022 \pm .007$	0
RQRN2E	580.4 ± 10.2	305.5 ± 7.4	130.5 ± 5.8	68.2 ± 6.2	$.024 \pm .018$	4.0 ± 0.2
RGRN	541.2 ± 8.1	288.0 ± 6.1	127.5 ± 5.2	96.8 ± 2.1	$.060 \pm .012$	0

Em síntese, os resultados de teste apresentados na Tabela 11 confirmam a superioridade da $\text{RQRN1E}_{\text{OPT}}$: além dos menores erros pontuais (RMSE $= 503 \pm 4$ e MAE $= 204 \pm 3$), apresenta a menor *Pinball Loss* (94 ± 3), cobertura alinhada à meta de 95 % e excelente calibração (0.014 ± 0.005), sem cruzamentos de quantis.

A injeção tardia de τ ($\text{RQRN1E}_{\tau_{\text{LATE}}}$) eleva a *Pinball* em apenas +1 %, mantendo a cobertura alvo (95.0 ± 1.8 %) e sofrendo apenas leve degradação de calibração (0.021 ± 0.006).

Inserir τ apenas na última camada (τ_{LAST}) compromete a forma dos quantis: *Pinball* cresce +3 %, a cobertura cai para 93.5 % e o erro de calibração sobe para 0.022.

Disponibilizar τ como entrada (τ_{INPUT}), mantendo a mesma arquitetura de rede, impõe gargalo severo: RMSE e MAE aumentam cerca de 14 %, *Pinball* dispara +43 %, cobertura perde mais de 6 p.p. e a calibração quase triplica, apontando limitação estrutural.

O arranjo em dois estágios (RQRN2E) é o pior cenário, com cobertura de apenas 68.2 %, erro de calibração próximo ao da variante τ_{last} e 4 % de cruzamentos.

5 Conclusão

Os experimentos conduzidos nos três *benchmarks* (*Yacht*, *Boston Housing* e *Diamonds*) mostram de forma consistente que o arranjo **RQRN1E$_{\text{opt}}$** —rede quantílica multitarefa de estágio único com inserção do τ otimizada e regularizações - supera todas as demais alternativas testadas (RGRN, demais variantes RQRN1E e o arranjo em dois estágios RQRN2E). A vantagem manifesta-se simultaneamente em:

- **Precisão pontual** — reduções médias de $\approx 10\,\%$ em RMSE e $12\,\%$ em MAE em relação à melhor concorrente por base, alcançando RMSE $= 0.65$ (*Yacht*), 1.70 (*Boston*) e 507 (*Diamonds*). Com uma diminuição de $30-60\,\%$ na *Pinball Loss* relativamente ao baseline gaussiano, mantendo cobertura média $94.8 \pm 0.9\,\%$ com calibração < 0.02.
- **Robustez monotônica** — nenhum cruzamento de quantis em todos os *folds* (Cruz.% $=$ 0).
- **Custo computacional** — tempo de treino $35-50\,\%$ inferior ao RQRN2E, mesmo após inclusão de SAM e busca Bayesiana de hiper–parâmetros.

Destacam-se três lições principais:

i) **Posicionamento de τ.** Inserir τ na penúltima camada fornece representações latentes ricas antes da restrição 1-Lipschitz final, atenuando o *dilution effect* observado em inserções precoces e evitando a perda de largura causada pelo caso τ_{input}.

ii) **Regularização espectral + SAM.** A combinação limitou a constante de Lipschitz, achou mínimos planos e estabilizou especialmente os quantis extremais do conjunto *Diamonds*, onde a heterocedasticidade é pronunciada.

iii) **Limite das hipóteses gaussianas.** A RGRN atinge coberturas próximas que permeiam o alvo mas ainda com calibrações precisões pontuais insatisfatórias, ilustrando a inadequação de modelos centrados em erros simétricos para problemas reais de precificação.

Trabalhos futuros. Os resultados sugerem cinco linhas promissoras:

P1. Estudos de simulação para avaliar o modelo RQRN1E$_{\text{opt}}$ em dados sintéticos.

P2. *Ensembles leves* (Deep Ensembles, SWAG, Sub-ensembles) para reduzir variância sem multiplicar o custo de treino.

P3. *NAS multi-objetivo* que otimize erro, calibração e latência simultaneamente.

P4. *Ordenação diferenciável* (SoftSort, NeuralSort) acoplada à perda Pinball para aprender quantis já livres de cruzamento.

P5. *Escalonamento massivo* via `bfloat16`, tensor sharding e FSDP, visando conjuntos com dezenas de milhões de exemplos.

Referências

AKIBA et al. Optuna: A next-generation hyperparameter optimization framework. In: *Proceedings of the 25th ACM SIGKDD Conference*. [S.l.: s.n.], 2019. p. 2623–2631.

AMORIM, W. E. R. de et al. Positional encoder graph quantile neural networks for geographic data. *arXiv preprint arXiv:2409.18865*, 2025.

ANIL et al. Sorting out lipschitz function approximation. In: *36th International Conference on Machine Learning (ICML)*. [S.l.: s.n.], 2019. p. 291–301.

ANIL, C. et al. Sorting out lipschitz function approximation. *Proceedings of the 36th International Conference on Machine Learning*, p. 291–301, 2019.

BARDENET et al. Collaborative hyperparameter tuning. *Proceedings of the 30th International Conference on Machine Learning*, p. 199–207, 2013.

BARTLETT, P. L. et al. Spectrally–normalized margin bounds for neural networks. In: *Advances in Neural Information Processing Systems (NeurIPS)*. [s.n.], 2017. p. 6240–6249. Disponível em: <https://arxiv.org/abs/1706.08498>.

BENGIO et al. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, v. 5, n. 2, p. 157–166, 1994.

BENGIO, Y. *Practical Recommendations for Gradient-Based Training of Deep Architectures*. 2012. ArXiv preprint arXiv:1206.5533.

BERGSTRA et al. Random search for hyper-parameter optimization. In: *Proceedings of the 28th International Conference on Machine Learning*. [S.l.: s.n.], 2011. p. 281–288.

BERGSTRA et al. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, v. 13, n. Feb, p. 281–305, 2012.

BERGSTRA, J.; BENGIO et al. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, v. 13, p. 281–305, 2012.

BISHOP, C. M. Training with noise is equivalent to tikhonov regularization. *Neural Computation*, v. 7, n. 1, p. 108–116, 1995.

BROCK, A. et al. High-performance large-scale training with mixed precision and bfloat16. *Proceedings of the 9th International Conference on Learning Representations*, 2021.

CANNON, A. J. Quantile regression neural networks: Implementation in r and application to precipitation downscaling. *Computers & Geosciences*, v. 37, n. 9, p. 1277–1284, 2011.

CASELLA, G.; BERGER, R. L. *Statistical Inference*. 2. ed. [S.l.]: Duxbury, 2002.

CISSé et al. Parseval networks: Improving robustness to adversarial examples. In: *International Conference on Machine Learning*. [S.l.: s.n.], 2017. p. 854–863.

DEVROYE, L. Non–uniform random variate generation. Springer, 1986.

EFRON et al. *An Introduction to the Bootstrap*. New York: Chapman & Hall/CRC, 1993.

FALKNER, S.; KLEIN, A.; HUTTER, F. Bohb: Robust and efficient hyperparameter optimization at scale. In: *Proceedings of the 35th International Conference on Machine Learning*. [S.l.: s.n.], 2018. p. 1436–1445.

FEURER, M.; HUTTER, F. Hyperparameter optimization. In: *Automated Machine Learning*. [S.l.]: Springer, 2019. p. 3–33.

FLOOD et al. Neural networks in civil engineering i: Principles and understanding. *ASCE Journal of Computing in Civil Engineering*, v. 8, n. 2, p. 131–148, 1994.

FORET, P. et al. Sharpness-aware minimization for efficiently improving generalization. In: *International Conference on Learning Representations*. [S.l.: s.n.], 2021.

FORET, P. et al. Sharpness-aware minimization for efficiently improving generalization. *Proceedings of the 8th International Conference on Learning Representations*, 2021.

FRAZIER, P. I. A tutorial on bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018.

GAN et al. Embedding based quantile regression neural network for probabilistic load forecasting. *Journal of Modern Power Systems and Clean Energy*, v. 6, n. 2, p. 244–254, 2018.

GÉRON, A. *Máos Øbra: Aprendizado de Máquina com Scikit-Learn, Keras e TensorFlow: Conceitos, Ferramentas e Técnicas Para a Construção de Sistemas Inteligentes*. 2. ed. [S.l.]: Alta Books, 2019.

GHARAT, S. O que, por que e qual? funções de ativação. *Medium*, 2019. Acessado em: 8 jul. 2025. Disponível em: <https://medium.com/@snehagharrat/o-que-por-que-e-qual-funcoes-de-ativacao-492b662b0232>.

GLOROT, X.; BENGIO, Y. Understanding the difficulty of training deep feedforward neural networks. p. 249–256, 2010.

GLOROT, X.; BENGIO, Y. Understanding the difficulty of training deep feedforward neural networks. In: *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS)*. [S.l.: s.n.], 2010. p. 249–256.

GOH et al. Back-propagation neural networks for modeling complex systems. *Artificial Intelligence in Engineering*, v. 9, n. 3, p. 143–151, 1995.

GOLUB, G. H.; LOAN, V. et al. *Matrix Computations*. 4. ed. [S.l.]: Johns Hopkins University Press, 2013.

GOODFELLOW et al. *Deep Learning*. [S.l.]: MIT Press, 2017.

HARRELL et al. A new distribution-free quantile estimator. *Biometrika*, v. 69, n. 3, p. 635–640, 1982.

HASTIEC et al. *The Elements of Statistical Learning: Data Mining, Inference and Prediction*. 2. ed. [S.l.]: Springer, 2009.

HE et al. Markov chain marginal bootstrap. *Journal of the American Statistical Association*, v. 97, p. 783–795, 2002.

HE et al. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. p. 1026–1034, 2015.

HENDRYCKS, D.; GIMPEL, K. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016. Acesso em: 01 jul. 2025.

JAMIESON, A. et al. Non-stochastic best arm identification and hyperparameter optimization. In: *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*. [S.l.: s.n.], 2016. p. 240–248.

JANTRE et al. Quantile regression neural networks: A bayesian approach. *Journal of Statistical Theory and Practice*, v. 15, n. 1, 2021.

JONES et al. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, v. 13, n. 4, p. 455–492, 1998.

KLEIN, A. et al. Fast bayesian optimization of machine learning hyperparameters on large datasets. In: *AISTATS*. [S.l.: s.n.], 2017.

KOENKER, R. *Quantile Regression*. 1st ed.. ed. Cambridge, UK: Cambridge University Press, 2005. v. 1. 349 p.

KOENKER, R.; BASSETT, G. Regression quantiles. *Econometrica*, v. 46, p. 33–50, 1978.

KULESHOV et al. Accurate uncertainties for deep learning using calibrated regression. In: *International Conference on Machine Learning*. [S.l.: s.n.], 2018. p. 2796–2804.

KULESHOV et al. Calibrated and sharp uncertainties in deep learning via density estimation. In: *International Conference on Machine Learning (ICML)*. [S.l.: s.n.], 2022. p. 11683–11693.

KULESHOV et al. Calibrated and sharp uncertainties in deep learning via density estimation. In: *Proceedings of the International Conference on Machine Learning (ICML)*. [S.l.: s.n.], 2022. p. 11683–11693.

LI et al. Hyperband: A novel bandit-based approach to hyperparameter optimization. In: *Proceedings of the 5th International Conference on Learning Representations*. [S.l.: s.n.], 2017.

LOSHCHILOV et al. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

LOSHCHILOV, I.; HUTTER, F. Decoupled weight decay regularization. *Proceedings of the 7th International Conference on Learning Representations*, 2019.

MCCULLAGH, P.; NELDER, J. A. *Generalized Linear Models*. 2. ed. London: Chapman & Hall, 1989. (Chapman & Hall/CRC Monographs on Statistics and Applied Probability).

MCCULLOCH et al. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, v. 5, n. 4, p. 115–133, 1943.

MERITY et al. Regularizing and optimizing lstm language models. *arXiv preprint arXiv:1708.02182*, 2017.

MIYATO, T. et al. Spectral normalization for generative adversarial networks. In: *International Conference on Learning Representations*. [S.l.: s.n.], 2018.

MOON et al. Learning multiple quantiles with neural networks. *Journal of Computational and Graphical Statistics*, v. 30, n. 4, p. 1238–1248, 2021.

NADEAU, C.; BENGIO, Y. Inference for the generalization error. *Machine Learning*, v. 52, n. 3, p. 239–281, 2003.

PAN et al. Study on the performance evaluation of online teaching using the quantile regression analysis and artificial neural network. *Journal of Supercomputing*, v. 72, n. 3, p. 789–803, 2016.

PASZKE, A. et al. Pytorch: An imperative style, high-performance deep learning library. In: *Advances in Neural Information Processing Systems*. [S.l.: s.n.], 2019. v. 32, p. 8024–8035.

PEDREGOSA et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, v. 12, p. 2825–2830, 2011.

PRADEEPKUMAR et al. Forecasting financial time series volatility using particle swarm optimization trained quantile regression neural network. In: *Institute for Development, Research in Banking Technology (IDRBT)*. Hyderabad, India: [s.n.], 2017. p. 1–14.

PRECHELT, L. Early stopping—but when? *Neural Networks: Tricks of the Trade*, p. 55–69, 1998.

RAMACHANDRAN et al. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017. Acesso em: 01 jul. 2025.

ROCKAFELLAR, R. T.; WETS, R. J.-B. *Variational Analysis*. [S.l.]: Springer, 1998.

RODRIGUES et al. Beyond expectation: Deep joint mean and quantile regression for spatiotemporal problems. *IEEE Transactions on Neural Networks and Learning Systems*, p. 1–14, 2020.

SAXE et al. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2014.

SHAHRIARI et al. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, v. 104, n. 1, p. 148–175, 2016.

SNOEK et al. Practical bayesian optimization of machine learning algorithms. In: *Advances in Neural Information Processing Systems*. [S.l.: s.n.], 2012. v. 25, p. 2951–2959.

SNOEK, J.; LAROCHELLE, H.; ADAMS, R. P. Practical bayesian optimization of machine learning algorithms. *Advances in Neural Information Processing Systems*, v. 25, p. 2951–2959, 2012.

SOBOL', I. M. The distribution of points in a cube and the approximate evaluation of integrals. *USSR Computational Mathematics and Mathematical Physics*, v. 7, n. 4, p. 86–112, 1967.

SRIVASTAVA et al. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, v. 15, n. 56, p. 1929–1958, 2014. Disponível em: <http://jmlr.org/papers/v15/srivastava14a.html>.

TAGASOVSKA et al. Single-target uncertainty prediction via deep ensembles. *arXiv preprint arXiv:1910.04848*, 2019.

TAKDIR, M. N. et al. A simple metric to quantify quantile crossing in multivariate conditional quantile estimation. In: *Proceedings of the 39th International Conference on Machine Learning (ICML 2022)*. [S.l.: s.n.], 2022.

WANG et al. Bfloat16: The secret to high-performance deep learning on cloud tpus. *Google Cloud Blog*, 2021. Accessed 30 Jun 2024.

WEHENKEL, A.; LOUPPE, G. Unconstrained monotonic neural networks. In: *Advances in Neural Information Processing Systems*. [S.l.: s.n.], 2019.

YOSHIDA, Y.; MIYATO, T. Spectral norm regularization for improving the generalizability of deep learning. In: *International Conference on Learning Representations (ICLR) – Workshop Track*. [s.n.], 2017. p. 1–15. Disponível em: <https://arxiv.org/abs/1705.10941>.

ZECO, E. S. Regressão quantílica multitarefa via redes neurais profundas. Dissertação de Mestrado em Estatística, Universidade de Brasília. 2024.