

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO CEARÁ
PRÓ-REITORIA DE PESQUISA, PÓS-GRADUAÇÃO E INOVAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Luiz Fernando Paulino Queiroz

**EXPLICAÇÕES BASEADAS EM LÓGICA PARA REDES NEURAIS
ARTIFICIAIS USANDO RELAXAMENTO PARCIAL DAS
CAMADAS COM FUNÇÃO DE ATIVAÇÃO RELU**

Fortaleza – CE

31 de outubro de 2025

Luiz Fernando Paulino Queiroz

EXPLICAÇÕES BASEADAS EM LÓGICA PARA REDES NEURAIS ARTIFICIAIS USANDO RELAXAMENTO PARCIAL DAS CAMADAS COM FUNÇÃO DE ATIVAÇÃO RELU

Dissertação submetida ao Programa de Pós-Graduação em Ciência da Computação do Instituto Federal de Educação, Ciência e Tecnologia do Ceará, como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Linha de Pesquisa: Inteligência Artificial

Orientador: Prof. Dr. Thiago Alves Rocha

Instituto Federal de Educação, Ciência e Tecnologia do Ceará (IFCE)

Pró-Reitoria de Pesquisa, Pós-Graduação e Inovação (PRPI)

Programa de Pós-Graduação em Ciência da Computação (PPGCC)

Fortaleza – CE

31 de outubro de 2025

Dados Internacionais de Catalogação na Publicação
Instituto Federal do Ceará - IFCE
Sistema de Bibliotecas - SIBI
Ficha catalográfica elaborada pelo SIBI/IFCE, com os dados fornecidos pelo(a) autor(a)

- Q3e Queiroz, Luiz Fernando Paulino Queiroz.
 EXPLICAÇÕES BASEADAS EM LÓGICA PARA REDES NEURAIAS ARTIFICIAIS USANDO
 RELAXAMENTO PARCIAL DAS CAMADAS COM FUNÇÃO DE ATIVAÇÃO RELU / Luiz Fernando
 Paulino Queiroz Queiroz. - 2025.
 85 f. : il. color.
- Dissertação (Mestrado) - Instituto Federal do Ceará, Mestrado em Ciência da Computação, Campus
 Fortaleza, 2025.
 Orientação: Prof. Dr. Thiago Alves Rocha.
1. Redes Neurais. 2. Inteligência Artificial Explicável. 3. Raciocínio Automatizado. 4. Programação
 Linear Inteira Mista. I. Título.

CDD 004



Instituto Federal de Educação, Ciência e Tecnologia do Ceará (IFCE)
Pró-Reitoria de Pesquisa, Pós-Graduação e Inovação (PRPI)
Programa de Pós-Graduação em Ciência da Computação (PPGCC)

Luiz Fernando Paulino Queiroz

Esta dissertação foi julgada adequada para a obtenção do título de Mestre em Ciência da Computação, sendo aprovada pela Coordenação do Programa de Pós-Graduação em Ciência da Computação do Instituto Federal de Educação, Ciência e Tecnologia do Ceará e pela banca examinadora:

Orientador: Prof. Dr. Thiago Alves Rocha

Instituto Federal de Educação, Ciência e Tecnologia do
Ceará (IFCE)

Prof. Dr. Carlos Henrique Leitão Cavalcante

Instituto Federal de Educação, Ciência e Tecnologia do
Ceará (IFCE)

Prof. Dr. Luis Henrique Bustamante de Moraes

Universidade Federal do Cariri (UFCA)

Fortaleza – CE

31 de outubro de 2025

*Este trabalho é dedicado às crianças adultas que,
quando pequenas, sonharam em se tornar cientistas.*

Agradecimentos

Os agradecimentos principais são direcionados à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior Brasil (CAPES), pelo apoio financeiro e institucional para realização deste mestrado. O presente trabalho foi realizado com apoio da CAPES Código de Financiamento 001.

Agradecimentos especiais são direcionados ao orientador que contribuiu para a formação do autor.

“A mente que se abre a uma nova ideia jamais voltará ao seu tamanho original.”
(Albert Einstein)

Resumo

As Redes Neurais Artificiais (RNAs) têm aplicações em diversas áreas. Porém, elas não podem ser facilmente interpretadas por seres humanos, por isso são classificadas como um algoritmo de caixa preta. Em sistemas sensíveis a falhas, como análise clínica de imagens e controle de voo, explicações para RNAs tornaram-se essenciais, motivando o desenvolvimento de métodos baseados em lógica para garantir interpretações confiáveis. Em vista disso, estudos recentes têm empregado modelos de Programação Linear Inteira Mista (MILP - Mixed Integer Linear Programming) para representar Redes Neurais Artificiais (RNAs) com funções de ativação ReLU por meio de formulações compostas por restrições lógicas expressas em termos de desigualdades lineares e variáveis binárias. Essa abordagem permite não apenas a verificação formal das decisões do modelo, mas também a geração de explicações minimais, em que os atributos identificados como relevantes são suficientes para justificar a predição realizada. No entanto, estas soluções sofrem problemas de escalabilidade. Assim, este trabalho foca em obter explicações mais rápidas em detrimento da minimalidade por meio da relaxação de algumas variáveis binárias que representam a função de ativação ReLU. Na condução de experimentos, realizamos uma comparação entre o método original sem relaxação e o método com relaxação, levando em consideração o tempo de execução e a quantidade de atributos nas explicações. Os resultados demonstram que a relaxação proporciona uma redução de até 59.9% no tempo médio de explicação para um aumento de apenas 10.32% na quantidade de atributos da explicação, mitigando o problema de escalabilidade em detrimento da minimalidade. Para trabalhos futuros sugerimos abordagens mais complexas na escolha das variáveis relaxadas.

Palavras-chave: Redes Neurais, Inteligência Artificial Explicável, Raciocínio Automatizado, Programação Linear Inteira Mista.

Abstract

Artificial Neural Networks (ANNs) have applications across various domains. However, they are not easily interpretable by humans and are therefore classified as black-box models. In safety-critical systems, such as clinical image analysis and flight control systems, the ability to explain ANN decisions has become essential. To address this, logic-based explanation methods have been developed to ensure correct interpretations of ANN predictions. Recent studies have employed Mixed Integer Linear Programming (MILP) to represent ANNs through formulations that combine logical constraints with linear inequalities and binary variables. This approach enables not only the formal verification of model decisions but also the generation of minimal explanations, in which all identified features are strictly necessary to justify the prediction. However, these solutions often face scalability issues. This work focuses on obtaining faster explanations by relaxing some of the binary variables that represent the ReLU activation function, trading minimality for computational efficiency. In our experiments, we compare the original (non-relaxed) method with the relaxed version, evaluating execution time and the number of attributes included in the explanations. The results demonstrate that relaxation provides a reduction of up to 23.6% in the time required to generate explanations with an increase of only 2.5% in the number of explanation attributes, mitigating the scalability problem at the expense of minimality. For future work, we suggest exploring more sophisticated strategies for selecting which variables to relax.

Keywords: Neural Networks, Explainable Artificial Intelligence, Automated Reasoning, Mixed-Integer Linear Programming.

Lista de ilustrações

Figura 1 – Grafo de uma RNA	32
Figura 2 – Processamento de um neurônio	33
Figura 3 – Incidência da função ReLU	35
Figura 4 – Função da ReLU	35
Figura 5 – Exemplo de modelagem	37
Figura 6 – Região factível com x_1 binário.	38
Figura 7 – Região factível com x_1 relaxado em $[0, 1]$	38
Figura 8 – Domínio do neurônio	42
Figura 9 – Exemplo de domínio	43
Figura 10 – Exemplo de previsão 1	46
Figura 11 – Exemplo de previsão 2	47
Figura 12 – Exemplo de previsão 3	48
Figura 13 – Domínio dos neurônios de uma RNA com ReLU	53
Figura 14 – Função da ReLU com relaxação	54
Figura 15 – Exemplo de saída sem relaxação	55
Figura 16 – Exemplo de saída sem relaxação que não modifica a classificação de saída	56
Figura 17 – Exemplo de saída com relaxação que modifica a classificação de saída	57
Figura 18 – Exemplo de saída com relaxação que não modifica a classificação de saída	58
Figura 19 – Gráfico do Tempo Médio de Explicação para o conjunto de dados Diabetes, utilizando diferentes configurações de relaxação aplicadas às redes neurais.	67
Figura 20 – Gráfico do Tamanho Médio de Explicação para o conjunto de dados Diabetes, utilizando diferentes configurações de relaxação aplicadas às redes neurais.	68
Figura 21 – Gráfico do Tempo Médio de Explicação para o conjunto de dados Glass, utilizando diferentes configurações de relaxação aplicadas às redes neurais.	69
Figura 22 – Gráfico do Tamanho Médio de Explicação para o conjunto de dados Glass, utilizando diferentes configurações de relaxação aplicadas às redes neurais.	69
Figura 23 – Gráfico do Tempo Médio de Explicação para o conjunto de dados Iris, utilizando diferentes configurações de relaxação aplicadas às redes neurais.	70

Figura 24 – Gráfico do Tamanho Médio de Explicação para o conjunto de dados Iris, utilizando diferentes configurações de relaxação aplicadas às redes neurais.	70
Figura 25 – Gráfico do Tempo Médio de Explicação para o conjunto de dados Wine, utilizando diferentes configurações de relaxação aplicadas às redes neurais.	71
Figura 26 – Gráfico do Tamanho Médio de Explicação para o conjunto de dados Wine, utilizando diferentes configurações de relaxação aplicadas às redes neurais.	71
Figura 27 – Gráfico do Tempo Médio de Explicação para o conjunto de dados Breast Cancer, utilizando diferentes configurações de relaxação aplicadas às redes neurais.	72
Figura 28 – Gráfico do Tamanho Médio de Explicação para o conjunto de dados Breast Cancer, utilizando diferentes configurações de relaxação aplicadas às redes neurais.	73
Figura 29 – Gráfico do Tempo Médio de Explicação para o conjunto de dados Digits, utilizando diferentes configurações de relaxação aplicadas às redes neurais.	73
Figura 30 – Gráfico do Tamanho Médio de Explicação para o conjunto de dados Digits, utilizando diferentes configurações de relaxação aplicadas às redes neurais.	74
Figura 31 – Gráfico do Tempo Médio de Explicação para o conjunto de dados Sonar, utilizando diferentes configurações de relaxação aplicadas às redes neurais.	75
Figura 32 – Gráfico do Tamanho Médio de Explicação para o conjunto de dados Sonar, utilizando diferentes configurações de relaxação aplicadas às redes neurais.	75
Figura 33 – Gráfico do Tempo Médio de Explicação para o conjunto de dados MNIST, utilizando diferentes configurações de relaxação aplicadas às redes neurais.	76
Figura 34 – Gráfico do Tamanho Médio de Explicação para o conjunto de dados MNIST, utilizando diferentes configurações de relaxação aplicadas às redes neurais.	77
Figura 35 – Exemplo de instância explicada retirada do conjunto MNIST.	78

Lista de tabelas

Tabela 1	–	Tempo Médio de Explicação ($\pm$ desvio padrão) por configuração dos conjuntos com menos de 30 atributos.	63
Tabela 2	–	Tamanho Médio de Explicação ($\pm$ desvio padrão) por configuração dos conjuntos com menos de 30 atributos.	64
Tabela 3	–	Tempo Médio de Explicação ($\pm$ desvio padrão) por configuração dos conjuntos com pelo menos 30 atributos. Em negrito, as configurações que lograram maior redução de tempo para o conjunto MNIST.	65
Tabela 4	–	Tamanho Médio de Explicação ($\pm$ desvio padrão) por configuração dos conjuntos com pelo menos 30 atributos.	66

Lista de Algoritmos

1	Explicação	49
---	----------------------	----

Lista de abreviaturas e siglas

MILP	Programação Linear Inteira Mista
ReLU	Unidade Linear Retificada
RNA	Rede Neural Artificial
TO	Tempo Esgotado

Lista de símbolos

$\in$	Pertence ao conjunto
$\notin$	Não pertence ao conjunto
$\wedge$	E lógico (conjunção)
$\vee$	Ou lógico (disjunção)
$\neg$	Negação lógica
$\Rightarrow$	Implica (consequência lógica)
$\Leftrightarrow$	Se e somente se
$=$	Igualdade
$\neq$	Diferente
$\geq$	Maior ou igual
$\leq$	Menor ou igual
$>$	Maior
$<$	Menor
$\forall$	Para todo
$\exists$	Existe
$\subseteq$	Subconjunto
$\cup$	União de conjuntos
$\cap$	Interseção de conjuntos
$\emptyset$	Conjunto vazio
$\models$	Consequência lógica (sintática ou semântica)
$\not\models$	Não é consequência lógica
$\setminus$	Diferença de conjuntos

Sumário

1	INTRODUÇÃO	27
1.1	Motivação	27
1.2	Descrição do problema	28
1.3	Objetivos Geral e Específicos	28
1.4	Produção científica	29
1.5	Estrutura da Dissertação	29
2	REVISÃO CONCEITUAL	31
2.1	Arquitetura das Redes Neurais	31
2.1.1	ReLU	34
2.2	Programação Linear Inteira Mista	35
2.2.1	Relaxação de Variáveis Binárias	36
2.3	Lógica de Primeira Ordem sobre Aritmética Linear Real	39
3	EXPLICAÇÕES BASEADAS EM LÓGICA PARA REDES NEURAIS ARTIFICIAIS	41
3.1	Codificação das RNAs utilizando MILP	41
3.1.1	Representando a RNA como um MILP	41
3.2	Explicação Tradicional das RNAs	45
4	RELAXAMENTO DAS VARIÁVEIS BINÁRIAS PARA FORNECER EXPLICAÇÕES	51
4.1	Codificação Relaxada	51
4.2	Explicação Relaxada	52
4.2.1	Interpretação dos Intervalos	53
5	EXPERIMENTOS	59
5.1	Ferramentas de Software	59
5.1.1	Hardware Utilizado	59
5.1.2	Repositórios dos Conjuntos de Dados	59
5.1.3	Treinamento das Redes Neurais Artificiais	61
5.2	Condução	61
5.3	Resultados	62
5.3.1	Resultados Gerais	62
5.3.2	Conjunto Diabetes	67
5.3.3	Conjunto Glass	68

5.3.4	Iris	69
5.3.5	Conjunto Wine	71
5.3.6	Conjunto Breast Cancer	72
5.3.7	Conjunto Digits	73
5.3.8	Conjunto Sonar	74
5.3.9	Conjunto MNIST	76
6	CONCLUSÃO	79
	REFERÊNCIAS	81

1 Introdução

Atualmente, as Redes Neurais Artificiais (RNAs) encontram aplicação em diversas áreas devido às suas funcionalidades, que permitem boas generalizações para problemas que não são facilmente resolvidos por algoritmos simples (HILL, 2016). Elas são úteis em campos como processamento de voz e imagem (LYASHENKO et al., 2021), análise de dados (*data mining*) (ASHA; KR, 2021), entre outros (LIU et al., 2025; ALGHOUL et al., 2018; MUSLEH et al., 2019; RAUBER, 2005; METZ, 2006; ABIODUN et al., 2018). Conforme apontado por Koh e Liang (2017), as Redes Neurais Artificiais (RNAs) são difíceis de interpretar por seres humanos, sendo classificadas como algoritmos de caixa preta. Nesse mesmo contexto, Ribeiro, Singh e Guestrin (2016) examinam a confiabilidade desses modelos em previsões e classificações, destacando os riscos associados ao seu uso em sistemas críticos que permitem apenas pequenas margens de erro. Além disso, vários estudos (CADAMURO; GILAD-BACHRACH; ZHU, 2016; WENG et al., 2018; ELBOHER; GOTTSCHLICH; KATZ, 2020) ressaltam a necessidade de desenvolver abordagens que possibilitem a interpretação e explicação dos algoritmos de *machine learning* e RNAs. Nesse sentido, muitos trabalhos têm se empenhado em criar métodos e ferramentas para interpretar e explicar o comportamento das RNAs (LIU et al., 2021).

1.1 Motivação

Pesquisas recentes (TJENG; XIAO; TEDRAKE, 2017; FISCHETTI; JO, 2018; VENZKE; CHATZIVASILEIADIS, 2020) produziram modelos de Programação Linear Inteira Mista (MILP - Mixed-Integer Linear Programming) que permitem verificar propriedades de RNAs de forma exata. Por exemplo, poderíamos verificar se um neurônio sempre tem saída zero, de forma que poderíamos eliminá-lo da rede. Ignatiev, Narodytska e Marques-Silva (2019) propuseram um algoritmo capaz de gerar explicações livres de redundância e com garantia formal de corretude, a partir de uma rede neural previamente treinada e de uma instância classificada. O algoritmo pode ser caracterizado como um método de explicação cujo objetivo é identificar subconjuntos minimais de atributos de entrada que são suficientes para manter a mesma predição da rede. Tais explicações são denominadas minimais, pois qualquer alteração em seus atributos inviabiliza a garantia da predição original. Trata-se de método adequado para modelar uma rede neural como um problema de lógica que pode ser resolvido por um solucionador de Programação Linear Inteira Mista. No entanto, estas soluções sofrem problemas de escalabilidade, uma vez que para verificar se as restrições lineares possuem solução é necessária uma busca robusta demandando muito tempo e poder de processamento, sendo considerado um problema

NP-Completo (KATZ et al., 2017).

O trabalho de Carvalho, Oliveira e Rocha (2024) aborda uma forma de otimizar o algoritmo proposto por Ignatiev, Narodytska e Marques-Silva (2019) por meio da remoção de restrições de indicação (BELOTTI et al., 2016). O trabalho foi capaz de melhorar o tempo de codificação e explicação. A codificação permite que o modelo esteja pronto para explicar qualquer exemplo do domínio, portanto pode ser realizado previamente, mas ainda está limitado a poucas camadas e a poucos neurônios por camadas.

Assim, para contornar o problema de escalabilidade, este trabalho foca aproveitar o modelo de representação de RNAs proposto em Carvalho, Oliveira e Rocha (2024) adicionando a relaxação de variáveis binárias para obter as explicações em tempo hábil. Essa abordagem reduz parcialmente a complexidade computacional ao lidar com variáveis binárias não relaxadas, mas compromete a minimalidade, gerando explicações ainda passíveis de simplificação.

1.2 Descrição do problema

Os algoritmos de explicação rigorosos (IGNATIEV, 2020) geralmente utilizam lógica e Programação Linear Inteira Mista (MARQUES-SILVA, 2023) para garantir formalmente a corretude. Portanto, estão limitados à complexidade exponencial dos solucionadores disponíveis (KOCH et al., 2022) devido a natureza do problema. Nesse caso, é possível notar que há uma forma de relaxar o modelo aproximando-o de um problema de programação linear, que possui complexidade polinomial (KARMARKAR, 1984). Assim, para contornar o problema de escalabilidade, este trabalho foca em utilizar o modelo de codificação proposto em Carvalho, Oliveira e Rocha (2024) para obter as explicações conforme o algoritmo de Ignatiev, Narodytska e Marques-Silva (2019), mas com a conversão de algumas variáveis binárias em variáveis contínuas. Essa abordagem tem como finalidade obter melhorias no desempenho do solucionador de Programação Linear Inteira Mista utilizando uma quantidade menor de variáveis binárias por camada, possibilitando uma diminuição no tempo necessário para realizar explicações e aumentando a capacidade de explicar RNAs maiores em tempo hábil.

1.3 Objetivos Geral e Específicos

Este trabalho tem como objetivo geral definir uma abordagem que permita gerar explicações modeladas conforme o algoritmo de Ignatiev, Narodytska e Marques-Silva (2019), mas reduzindo o tempo de execução. Neste trabalho vamos utilizar a modelagem de Carvalho, Oliveira e Rocha (2024). Nossa abordagem consiste do relaxamento de variáveis binárias da modelagem.

Em seu sentido mais particular, os objetivos específicos são:

- Apresentar as modificações necessárias para gerar explicações relaxadas;
- Elucidar as consequências das modificações causadas pelo relaxamento;
- Realizar experimentos comparativos para avaliar as diferenças no tempo e no tamanho das explicações.

1.4 Produção científica

Durante este projeto de mestrado, os seguintes trabalhos científicos foram aceitos e publicados, a saber:

- **Gomes, R., Ribeiro, J., Queiroz, L., Rocha, T. A., 2025. Bound Propagation meets Constraint Simplification: Improving Logic-based XAI for Neural Networks.** In: *Proceedings of the International Joint Conference on Neural Networks (IJCNN 2025)*. Roma, Itália. Aceito para publicação.
- **Queiroz, L. F. P., Cavalcante, C. H. L., Rocha, T. A., 2025. Slice and Explain: Logic-Based Explanations for Neural Networks through Domain Slicing.** In: *Proceedings of the Brazilian Conference on Intelligent Systems (BRACIS 2025)*. Fortaleza, Ceará, Brasil. Aceito para publicação.

1.5 Estrutura da Dissertação

A estrutura dessa dissertação seguirá com o capítulo 2 descrevendo o estado da arte dos conceitos necessários para produzir Redes Neurais Artificiais e gerar explicações. O capítulo 3 possui tópicos voltados para descrever como as RNAs utilizadas nesse trabalho podem ser modeladas via MILP, como modificações chamadas de simplificação podem melhorar o desempenho do solucionador e como explicações minimais podem ser geradas. Por sua vez, o Capítulo 4 explora a estratégia de modificação de variáveis no modelo original, com o objetivo de aprimorar o desempenho computacional do solucionador. Tal abordagem preserva a corretude da solução, ainda que à custa da perda de minimalidade na formulação. Em seguida, o Capítulo 5 apresenta os experimentos realizados, contemplando, mas não se limitando a: os conjuntos de dados empregados, as arquiteturas de redes neurais artificiais (RNAs) treinadas e as métricas utilizadas para avaliação das explicações geradas. O capítulo discute quantitativamente os resultados numéricos e analisa qualitativamente as propriedades das explicações geradas. Por fim, o Capítulo 6 expõe as principais repercussões notáveis a partir dos resultados, discutindo as aplicações compatíveis com a abordagem. Também é discutido o que pode ser abordado em trabalhos futuros.

2 Revisão Conceitual

Este capítulo aborda alguns conceitos estritamente necessários para contextualização do assunto central dessa dissertação.

2.1 Arquitetura das Redes Neurais

A estrutura das RNAs segue algumas regras. Possuem neurônios, camadas e conexões entre neurônios. Os neurônios são dispostos em camadas geralmente referenciadas por um índice. Uma rede neural pode possuir várias camadas, e as camadas intermediárias (que não são de entrada nem de saída) são chamadas de camadas ocultas. Cada camada adicionada resulta em mais conexões. A representação visual desse modelo pode ser visto na Figura 1. O objetivo da rede é computar uma entrada e gerar uma saída que representa uma previsão de um comportamento (HUANG; CHEN; BABRI, 2000).

Uma RNA precisa passar por uma fase de treinamento para refinar sua capacidade de prever comportamentos. Esse treinamento envolve testar a precisão da RNA utilizando uma base de dados e algoritmos de aprendizado como a retropropagação (LEUNG; HAYKIN, 1991) para calcular as mudanças a serem aplicadas nos pesos dos neurônios. Porém, uma RNA só precisa passar por esse processo uma vez, depois ela já pode ser considerada treinada.

Nos modelos de RNAs que iremos abordar, os neurônios de uma determinada camada não podem estar conectados aos neurônios da mesma camada, além disso não podem enviar informações para neurônios de camadas anteriores. Dessa forma, a propagação do sinal apenas acontece entre camadas em um sentido. Esses tipos de RNAs são conhecidas pela nomenclatura *feedforward*, ou seja, a informação não retrocede na rede (ELDAN; SHAMIR, 2016).

Uma forma interessante de visualizar uma RNA *feedforward* é como um grafo direcionado acíclico (YOU et al., 2020). Dessa forma, podemos notar que toda a informação que parte da camada de entrada da RNA só flui em uma direção, até chegar finalmente em algum neurônio da camada de saída. Cada nó do grafo representa um neurônio armazenando, processando e passando informação adiante.

Os neurônios da primeira camada formam a entrada da RNA e a sua quantidade é definida pela quantidade de atributos definidos para o problema. A última camada representa a saída da rede e a sua quantidade pode variar de acordo com a forma escolhida para resolver o problema. As camadas intermediárias são chamadas de camadas ocultas e podem existir em qualquer quantidade, além de comportar qualquer quantidade de

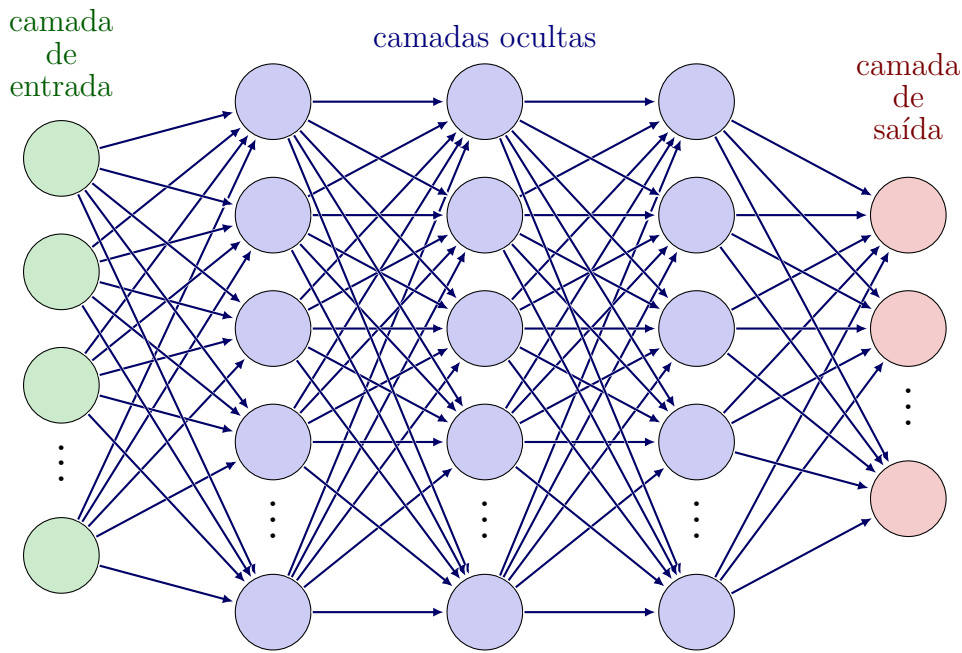


Figura 1 – Grafo de uma RNA

Fonte: autoral

neurônios. Assim, a quantidade pode ser ajustada para oferecer melhor eficiência para cada problema. As camadas ocultas representam o aprendizado e o processamento, sua função é receber o conjunto de valores de saída da camada anterior e gerar outra saída para que o conjunto de valores seja propagado.

Como as RNAs são capazes de resolver vários tipos de problemas, a estrutura de saída também pode variar em relação a diversas possibilidades como a quantidade de neurônios de saída e a representação dos valores de saída (ROCHA; CORTEZ; NEVES, 2007). Nesse trabalho serão abordadas apenas redes de classificação, na qual a saída necessariamente deve ter dois ou mais neurônios. Nos problemas de classificação, como todos os neurônios de saída são ativados, uma previsão pode ser obtida através do valor máximo, ou seja, o neurônio de saída que tiver o maior valor em relação a todos os outros. Esse neurônio representará a classe que a rede previu como sendo o correto em relação à entrada que ela recebeu.

A função de cada neurônio é realizar uma soma ponderada, ou seja, cada neurônio deve captar todos os valores que receber e multiplicar pelo peso associado àquele valor. O resultado deve ser somado a um valor chamado viés e logo após deve ser aplicado em uma função de ativação e passado adiante até que todos os sinais cheguem à saída.

Em uma RNA, podemos definir o valor da soma ponderada do neurônio de acordo com o modelo na Figura 2. Para esta equação, $x_i^{(j)}$ representa o valor do neurônio i na camada j . A notação $p_{i,j,k}$ representa o valor do peso que será multiplicado pelo valor do neurônio i da camada j e o resultado será computado junto aos outros no neurônio k da

camada $j + 1$, ou seja, o sinal sairá de $x_i^{(j)}$ e chegará em $x_k^{(j+1)}$. Já o viés $v_k^{(j+1)}$ é uma constante que será somada ao valor da soma ponderada para computar o resultado do valor do neurônio $x_k^{(j+1)}$. O símbolo σ é a representação da função de ativação que será detalhada no final dessa seção.

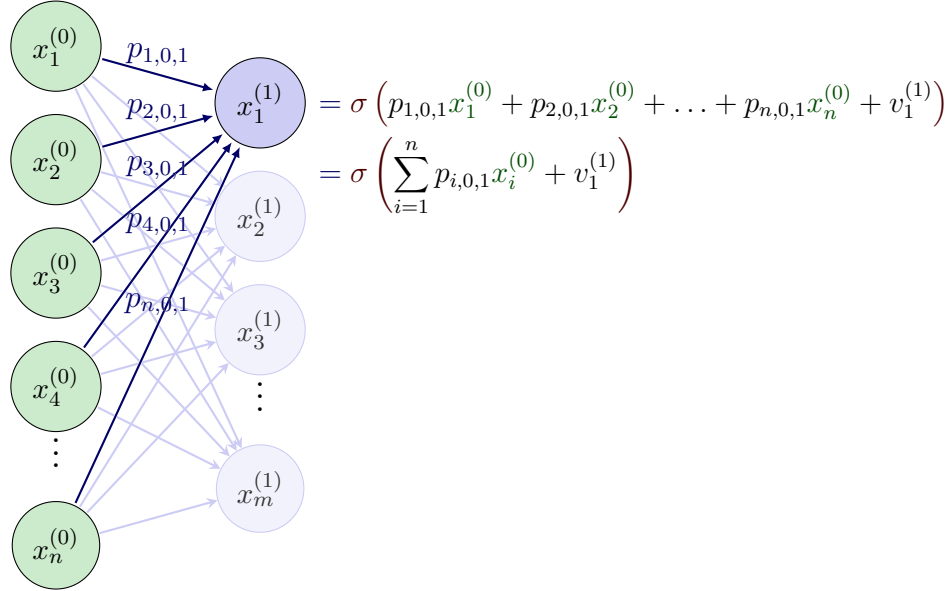


Figura 2 – Processamento de um neurônio

Fonte: autoral

Também é possível evidenciar os valores dos neurônios de uma camada através de uma multiplicação de matrizes e uma soma, na qual as matrizes multiplicadas tem relação com os pesos dos neurônios e os respectivos valores que eles computaram e a matriz resultado é somada com a matriz que representa o respectivo viés de cada neurônio.

$$\begin{pmatrix} x_1^{(k)} \\ x_2^{(k)} \\ \vdots \\ x_m^{(k)} \end{pmatrix} = \sigma \left[\begin{pmatrix} p_{1,k-1,1} & p_{2,k-1,1} & \dots & p_{n,k-1,1} \\ p_{1,k-1,2} & p_{2,k-1,2} & \dots & p_{n,k-1,2} \\ \vdots & \vdots & \ddots & \vdots \\ p_{1,k-1,m} & p_{2,k-1,m} & \dots & p_{n,k-1,m} \end{pmatrix} \begin{pmatrix} x_1^{(k-1)} \\ x_2^{(k-1)} \\ \vdots \\ x_n^{(k-1)} \end{pmatrix} + \begin{pmatrix} v_1^{(k)} \\ v_2^{(k)} \\ \vdots \\ v_m^{(k)} \end{pmatrix} \right] \quad (2.1)$$

Na equação 2.1, x_i^k é o neurônio i da camada k . Já o elemento $p_{i,k-1,l}$ representa o fator peso no qual: i é o índice do neurônio da camada de origem do sinal; $k - 1$ é o índice da camada de origem; l é o índice do neurônio da camada de destino do sinal; e k representa o índice da camada de destino. Já o $v_l^{(k)}$ representa o viés de cada neurônio l na camada de destino k .

Dessa forma, a matriz representa a informação de todos os valores, pesos, e vieses entre as camadas k e $k - 1$, na qual o sinal que parte da $k - 1$ chegará na camada k , como pode ser visto na equação 2.2.

$$x^{(k)} = \sigma \left(\mathbf{P}^{(k-1)} x^{(k-1)} + \mathbf{v}^{(k-1)} \right) \quad (2.2)$$

Os neurônios $x^{(k)}$ são definidos pela camada $k - 1$, que é a camada de origem do sinal, e a camada k é o destino do sinal.

Antes que os valores cheguem no próximo neurônio, geralmente é implementada uma função de ativação que vamos representar por σ . Ela permite que pequenas alterações ocorram antes que o valor seja entregue ao próximo neurônio, influenciando diretamente no resultado da saída. Por isso, geralmente são utilizadas diretamente após a soma ponderada.

Existem diversos tipos de funções de ativação, mas a função de ativação ReLU, uma das mais utilizadas em redes neurais (MERCIONI; HOLBAN, 2020), é a única usada neste trabalho por ser facilmente representada por restrições lineares em modelos de Programação Linear Inteira Mista.

2.1.1 ReLU

A ReLU (Rectified Linear Unit) produz resultados maiores ou iguais a zero. A principal vantagem de usar a função ReLU sobre outras funções de ativação é que ela pode desativar neurônios durante a propagação do sinal. Isso acontece porque a função ReLU retorna 0 para todas as entradas negativas, e o próprio valor para entradas não negativas, então ela tende a “apagar” alguns neurônios durante o treinamento, aumentando a capacidade de modelar a predição.

Na Figura 3 podemos notar como funciona a utilização da função ReLU ao longo da RNA. Ela recebe como parâmetro o valor que a função de soma ponderada retorna. O cálculo do resultado após a ativação de uma ReLU pode ser encontrado através de uma simples equação:

$$ReLU(x) = \max(0, x) = \begin{cases} x, & \text{se } x \geq 0 \\ 0, & \text{se } x < 0 \end{cases} \quad (2.3)$$

Na qual $ReLU(x)$ representa a saída do neurônio, $\max()$ é uma função que retorna o maior dos valores recebidos no parâmetro e x representa a soma ponderada que o neurônio obteve. A Figura 4 demonstra o formato linear e descontínuo que a função ReLU possui. A função de ativação é implementada em todos os neurônios de uma RNA, com exceção dos neurônios de entrada e saída, que assim podem registrar valores potencialmente no intervalo $(-\infty, +\infty)$.

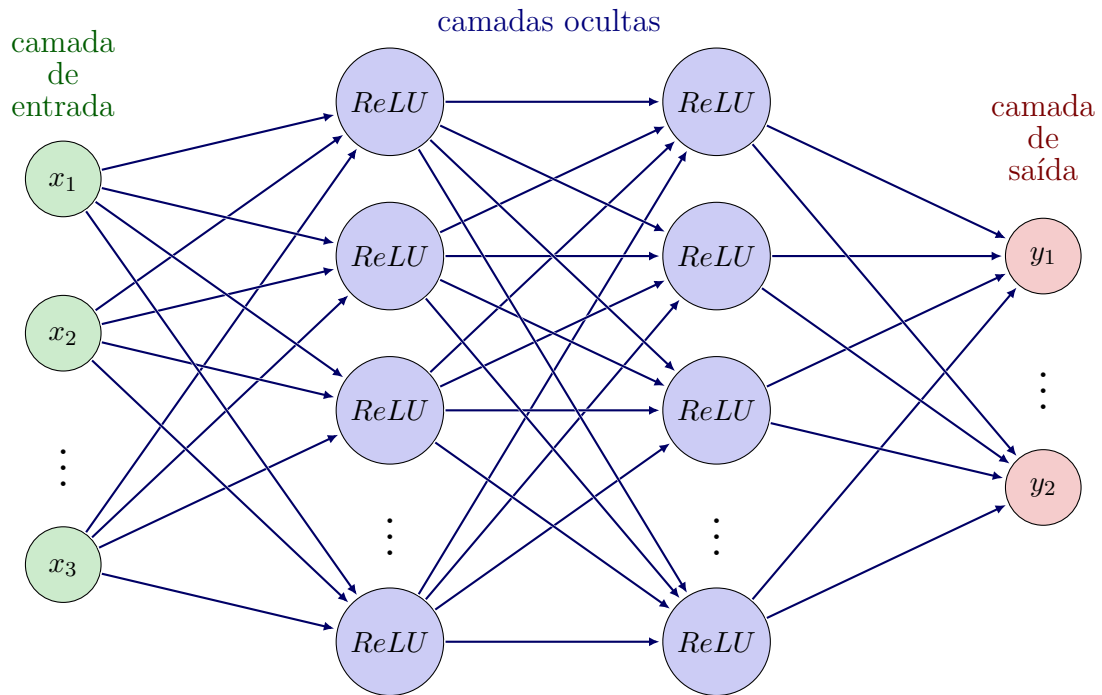


Figura 3 – Incidência da função ReLU

Fonte: autoral

$$ReLU(x) = \max(0, x)$$

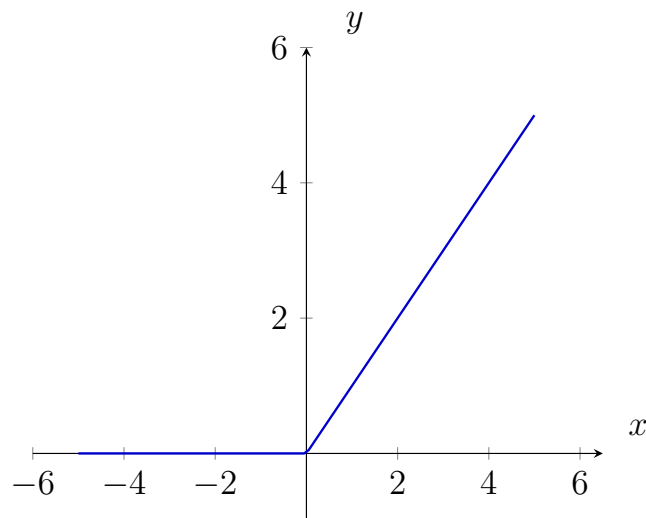


Figura 4 – Função da ReLU

Fonte: autoral

2.2 Programação Linear Inteira Mista

A Programação Linear Inteira Mista (MILP - Mixed Integer Linear Programming) é uma generalização da Programação Linear Inteira, na qual resolve problemas cuja essência pode ser representada por meio de geometria (BEST; RITTER, 1985). Um modelo MILP é capaz de encontrar soluções para problemas em que algumas ou todas as variáveis

pertencem ao conjunto dos números inteiros (BÉNICHOU et al., 1971).

Uma solução é encontrada através da modelagem de um problema. As variáveis representam as dimensões do polígono que será criado, portanto precisam ser limitadas. Assim, devem ser inseridas restrições limitando os valores das variáveis. As restrições, geometricamente, vão criando o polígono que representa o espaço geométrico na qual uma solução pode ser encontrada.

O polígono pode ser representado de forma visual, quando tem até três dimensões. Acima disso, é mais difícil criar figuras que o representem de maneira nítida e com o formato adequado. As equações 2.4, 2.5, 2.6 e 2.7 são exemplos de como uma modelagem pode ser feita.

$$\max \quad x_1 + x_2 \quad (2.4)$$

$$\text{s.t.} \quad x_1 + x_2 \leq 12 \quad (2.5)$$

$$x_1 - x_2 \geq 0 \quad (2.6)$$

$$x_1, x_2 \geq 0 \quad (2.7)$$

Na primeira linha, são declaradas as variáveis a serem maximizadas ou minimizadas. Por exemplo, x_1 pode representar a insulina de uma pessoa, maximizá-la seria encontrar o maior valor e minimizá-la seria encontrar o menor valor. Analogamente podemos descrever x_2 e supor que ela defina a glicose da pessoa. Nas linhas subsequentes, são inseridas as restrições que definem o domínio para o problema. O gráfico da Figura 5 demonstra visualmente o efeito de construir uma representação geométrica da modelagem anterior. Na Figura 5, a linha vermelha traçada no ponto abaixo da linha verde é chamada de ponto ótimo e é a solução que melhor corresponde à equação (2.4). Caso o problema não precise ter uma solução ótima, a sombra em azul delimita a chamada região factível, que é uma região que satisfaz todas as restrições.

Essa capacidade de solucionar problemas lineares auxilia na construção de uma codificação que permite mapear os limites inferiores e superiores de valores que um neurônio de uma RNA pode atingir. Dessa forma, podemos calcular os limites de todos os neurônios e formular um modelo contido que representa a rede geometricamente.

Essa contenção é útil para mapear todo o comportamento da RNA em relação aos dados usados para treiná-la e às previsões que ela realiza. Assim, é possível adicionar e remover restrições para que inferências, interpretações e explicações possam ser realizadas.

2.2.1 Relaxação de Variáveis Binárias

Programas de Programação Linear Inteira Mista (MILP) frequentemente utilizam variáveis binárias, também denominadas variáveis de decisão, que introduzem alterações

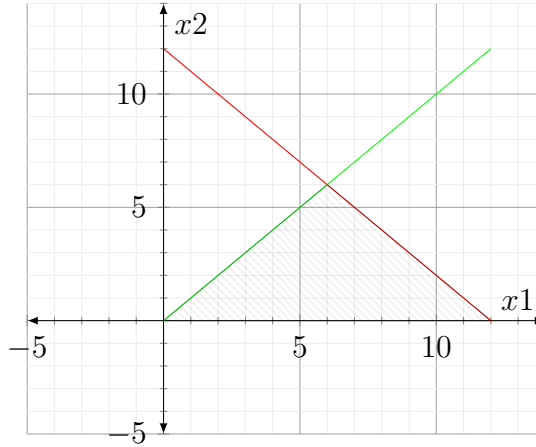


Figura 5 – Exemplo de modelagem

Fonte: autoral

abruptas no espaço da solução factível. Essas variáveis binárias, um caso especial de variáveis inteiras restritas ao conjunto $\{0, 1\}$, provocam uma redução e descontinuidade na região factível do problema. A reformulação dessas variáveis binárias para variáveis contínuas (PIA; KHAJAVIRAD, 2025), restritas ao intervalo $[0, 1]$, configura uma relaxação amplamente empregada como etapa intermediária em algoritmos solucionadores, especialmente na aplicação da técnica *branch-and-bound* (SMITH; TASKIN, 2008). A seguir, apresenta-se uma ilustração que demonstra o efeito dessa relaxação sobre a região factível.

As equações 2.8, 2.9, 2.10 representam as restrições de um problema com x_1 sendo uma variável binária não relaxada e x_2 sendo uma variável contínua.

$$x_1 + x_2 \geq 3 \quad (2.8)$$

$$x_2 \geq 2 \quad (2.9)$$

$$x_1 \in \{0, 1\}, \quad x_2 \in \mathbb{R} \quad (2.10)$$

Na Figura 6, observa-se a região factível do problema original, na qual a linha vermelha representa a desigualdade definida pela equação 2.8 e a linha verde corresponde à desigualdade da equação 2.9. A região factível está restrita às linhas verticais destacadas em preto, refletindo a natureza discreta da variável considerada.

Em contrapartida, a Figura 7 ilustra a expansão da região factível decorrente da relaxação da variável inteira para contínua, evidenciada pela área sombreada em cinza que se estende por todo o plano. Essa relaxação é formalmente realizada pela substituição da equação 2.10 pela equação 2.11, ampliando o domínio das variáveis e possibilitando soluções mais flexíveis no espaço contínuo.

$$x_1 \in [0, 1], \quad x_2 \in \mathbb{R} \quad (2.11)$$

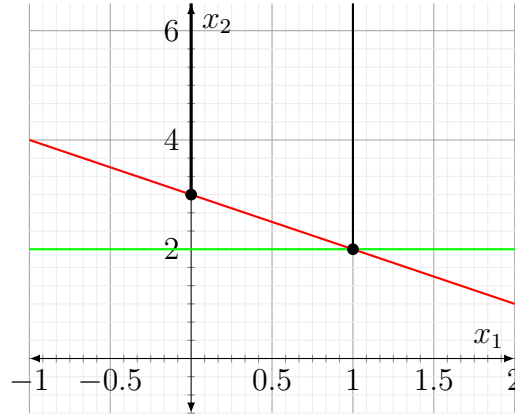


Figura 6 – Região factível com x_1 binário.

Fonte: autoral

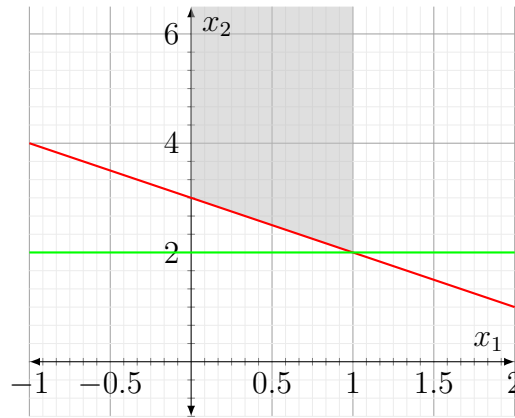


Figura 7 – Região factível com x_1 relaxado em $[0, 1]$.

Fonte: autoral

A relaxação, por si só, modifica estruturalmente o problema original, podendo, em muitos casos, constituir uma barreira para a obtenção de soluções aceitáveis. Isso ocorre porque a simples aplicação da relaxação pode alterar a solução ótima do modelo original. Embora essa técnica seja reconhecida por aproximar o problema de uma solução computacionalmente tratável, geralmente em tempo polinomial (ALEMANY; KASPRZYK; MAGNAGO, 2018), é comum a necessidade de incorporar restrições adicionais com o intuito de recuperar características desejáveis da formulação original e restabelecer as soluções previamente observadas (LIBERTI, 2021). No contexto de modelagens MILP aplicadas a redes neurais para verificação formal, a relaxação de variáveis já foi explorada (ANDERSON et al., 2020) com a inclusão de restrições complementares que visam mitigar os efeitos indesejáveis da flexibilização das variáveis inteiras. No entanto, estas restrições adicionais modificam a complexidade do problema e não serão necessárias, pois apenas a relaxação é suficiente para garantir a corretude de explicações, conforme a seção 4.1 expõe.

2.3 Lógica de Primeira Ordem sobre Aritmética Linear Real

A computação de explicações com garantia de corretude aproveita-se de alguns conceitos de lógica de primeira ordem (FOL - First Order Logic) (KROENING; STRICHMAN, 2016) e de satisfabilidade (BIERE; HEULE; MAAREN, 2009). Com esses conceitos, é possível empregar fórmulas de primeira ordem sem quantificadores sobre a teoria da aritmética linear real (LRA). As variáveis podem assumir valores no conjunto dos números reais $\mathbb{R}$. São consideradas fórmulas da forma da equação 2.12, em que $n \geq 1$ é um número natural, cada w_i e b são constantes reais fixas, e cada x_i é uma variável de primeira ordem. Usualmente, se F e G são fórmulas, então $(F \wedge G)$, $(F \vee G)$, $(\neg F)$ e $(F \rightarrow G)$ também são fórmulas.

$$\sum_{i=1}^n w_i x_i \leq b \quad (2.12)$$

É permitido o uso de outras letras para variáveis em vez de x_i , tais como s_i , z_i , etc. Por exemplo, a expressão da equação 2.13 é uma fórmula, de acordo com essa definição. Observe que também é permitido realizar abreviações padrão como $-(2.5x_1 + 3.1s_2 < 6)$ para $2.5x_1 + 3.1s_2 \geq 6$.

$$(2.5x_1 + 3.1s_2 \geq 6) \wedge (x_1 = 1 \vee x_1 = 2) \wedge (x_1 = 2 \rightarrow s_2 \leq 1.1) \quad (2.13)$$

Como assumimos a semântica das fórmulas sobre o domínio dos números reais, uma *atribuição* A para uma fórmula F é uma função que associa variáveis de primeira ordem a elementos do domínio dos números reais. Uma atribuição A satisfaz uma fórmula F se F for verdadeira sob essa atribuição. Por exemplo, a atribuição $\{x_1 \mapsto 2.3, s_2 \mapsto 1\}$ satisfaz a fórmula da equação 2.13. Por outro lado, a atribuição $x_1 \mapsto 2, s_2 \mapsto 1.05$ não satisfaz a fórmula. Além disso, uma atribuição satisfaz uma fórmula se e somente se ela torna a fórmula verdadeira. Uma fórmula F é dita satisfatível se existir uma atribuição que a satisfaça. Por exemplo, a fórmula da equação 2.14 é insatisfatível, já que nenhuma atribuição satisfaz ambas as desigualdades simultaneamente.

$$(x_1 \geq 2) \wedge (x_1 \leq 1) \quad (2.14)$$

Dado um conjunto de fórmulas Γ e uma fórmula G , dizemos que $\Gamma \models G$ se toda atribuição que satisfaz todas as fórmulas de Γ também satisfaz G . Como exemplo ilustrativo, seja $\Gamma = \{x_1 = 2, x_2 \geq 1\}$ e $G = x_1 + x_2 \geq 2$. Então, $\Gamma \models G$.

Esse conceito de consequência lógica é central para garantir a correção das explicações baseadas em competição, como explorado recentemente em (IGNATIEV; NA-

RODYTSKA; MARQUES-SILVA, 2019). Ele também fornece uma ferramenta fundamental para garantir transparência em modelos preditivos baseados em redes neurais.

Verificar satisfatibilidade e consequência lógica é uma tarefa fundamental na lógica formal (SHAPIRO; KISSEL, 2018). É sabido que, para todos os conjuntos de fórmulas Γ e toda fórmula G , temos que:

$$\Gamma \models G \text{ se, e somente se, } \Gamma \cup \{\neg G\} \text{ é insatisfatível.} \quad (2.15)$$

Por exemplo, $\{x_1 = 2, x_2 \geq 1 \neg((2.5x_1 + x_2 \geq 5) \wedge (x_1 = 1 \vee x_2 = 2))\}$ não possui nenhuma atribuição satisfatória, já que toda atribuição que satisfaz uma fórmula não satisfaz a outra. Assim, $(2.5x_1 + x_2 \geq 5) \wedge (x_1 = 1 \vee x_2 = 2)$ é uma consequência lógica de $\{x_1 = 2, x_2 \geq 1\}$. Dado que a abordagem desse trabalho se baseia no conceito de consequência lógica, é possível explorar essa conexão no contexto de explicações para redes neurais mesmo com relaxações de variáveis conceituadas na seção anterior.

3 Explicações Baseadas em Lógica para Redes Neurais Artificiais

Este capítulo apresenta modelos de representação e explicação de redes neurais já presentes na literatura, retomando uma parte da revisão bibliográfica já feita no capítulo anterior.

3.1 Codificação das RNAs utilizando MILP

Partindo da premissa da existência de uma RNA já treinada, podemos extrair suas informações, ou seja, os pesos e vieses que são importantes para a estrutura específica da rede. Em seguida podemos codificá-la na forma de restrições lineares, uma vez que os neurônios têm a função de realizar somas ponderadas.

3.1.1 Representando a RNA como um MILP

Após extrairmos os dados essenciais da RNA (pesos e vieses), podemos implementar um modelo de Programação Linear Inteira Mista (MILP) que represente fidedignamente o comportamento de predições dela. O modelo de representação em MILP deste trabalho tem como base a implementação de Tjeng, Xiao e Tedrake (2017), na qual formula a RNA através do domínio dos dados de entrada.

Para a representação, é essencial que todos os valores possíveis admitidos pelos neurônios sejam conhecidos, então estamos considerando que cada atributo tem um domínio conhecido. Por exemplo, uma restrição presente na realidade é que a idade de uma pessoa não pode ser menor do que 0 anos e provavelmente não será maior do que 130. Portanto, os valores que os neurônios podem ter dependem apenas do domínio dos atributos.

Nesse contexto, os intervalos de valores possíveis de cada neurônio da RNA serão calculados através de iterações em cada camada da RNA. Dessa forma, a camada de saída da codificação será capaz de representar o intervalo de valores de saída em relação ao domínio dos atributos daquela RNA.

Para obter todos os intervalos, podemos associar cada um dos neurônios a um problema de maximização e um de minimização representando os limites no qual o valor de saída de um neurônio pode apresentar, e as restrições são obtidas através da equação entre valores das camadas anteriores, conforme a Figura 8. Observe que $Min(x_1^{(1)})$ representa o menor valor possível que o respectivo neurônio pode assumir, assim como o $Max(x_1^{(1)})$

representa o maior valor. $D_1^{(1)}$ representa o domínio do neurônio $x_1^{(1)}$.

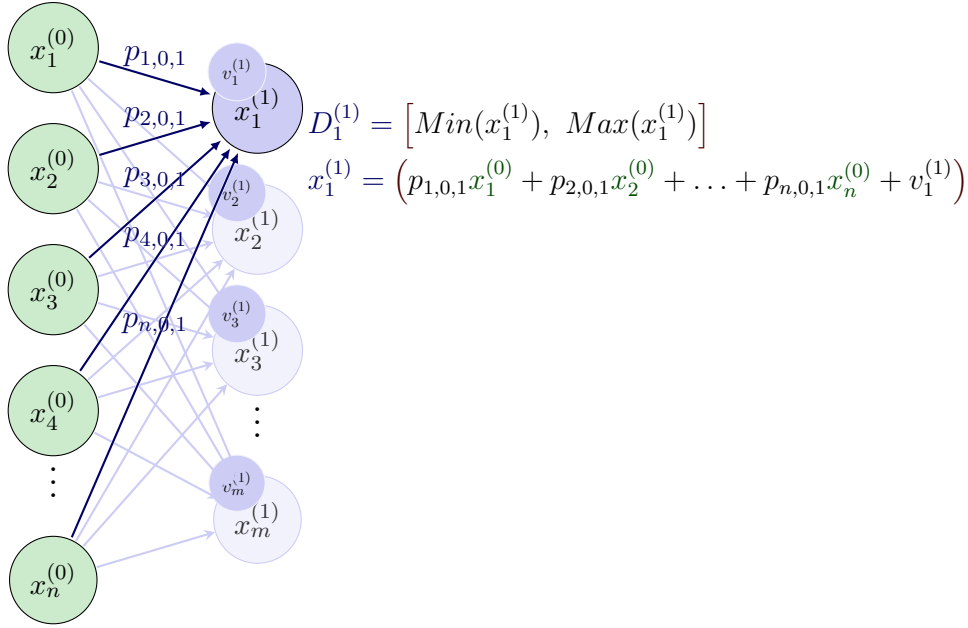


Figura 8 – Domínio do neurônio

Fonte: autoral

Para detalhar melhor, podemos começar pelos neurônios da primeira camada, que possuem um domínio. Cada entrada j possui domínio $D_j^{(0)} = [li_j^0, ls_j^0]$, na qual li_j^0 e ls_j^0 denotam, respectivamente, os valores mínimo e máximo que aquela entrada pode assumir.

Por exemplo, se retomarmos à informação do exemplo anterior, na qual a idade mínima é de 0 anos e a idade máxima é de 130 anos, então uma RNA construída para realizar predições com base nessa informação poderá ser modelada com a variável $x_1^{(0)}$ representando idade com li_1^0 igual a 0 e ls_1^0 igual a 130. Da mesma forma, podemos supor um atributo altura representado por uma variável x_2^0 , que possui os valores mínimo e máximo de 130 e 220 centímetros, respectivamente.

A Figura 9 ilustra uma parte da modelagem para obter o domínio de um neurônio primeira camada intermediária. Para o exemplo, estão inclusos os pesos e viés associados ao neurônio da camada a ser calculada. Estão fixos os valores de peso em 1.2 e -0.2 , e o viés em 24. Dessa forma, é possível perceber que os valores mínimo e máximo não são tão intuitivos, pois maximizar um neurônio da próxima camada não é equivalente a maximizar todas as entradas. Ao obter o domínio do neurônio, há garantia de que não há valores válidos para aquela RNA fora dele. Assim, de início podemos adicionar todas as variáveis de entrada à modelagem com as respectivas restrições de domínio.

Em seguida, podemos calcular os limites dos valores assumidos pelos neurônios da próxima camada (primeira camada intermediária). Por exemplo, o valor de $x_1^{(1)}$ está limitado ao intervalo $[-20, 154]$. Esse cálculo pode ser obtido através de uma modelagem intermediária, parecendo, intuitivamente, como uma montagem. Em toda a camada,

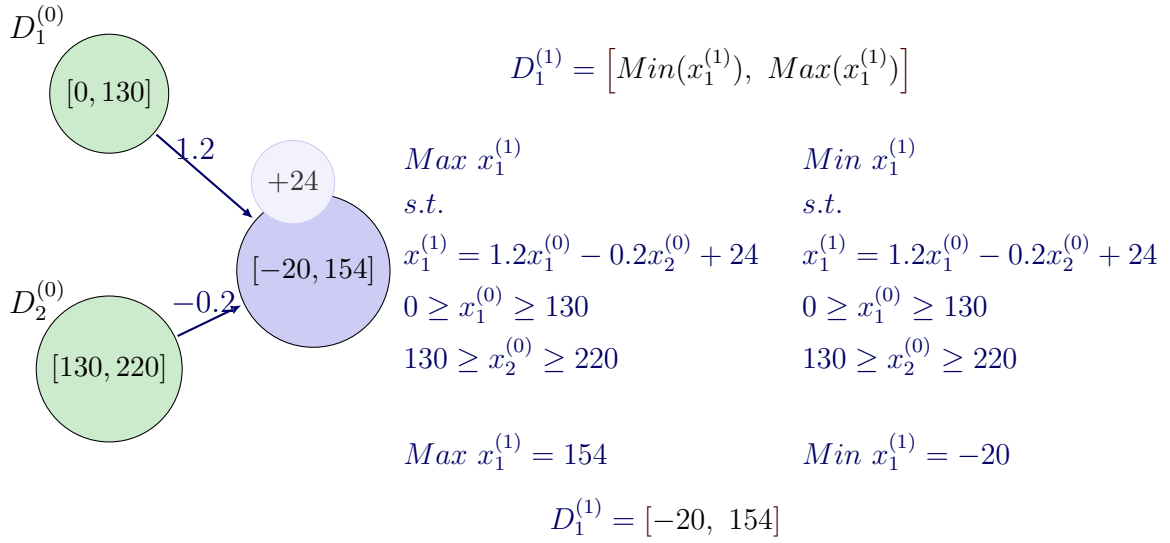


Figura 9 – Exemplo de domínio

Fonte: autoral

iremos iterar por cada um dos neurônios, realizando uma solução de maximização e minimização da soma ponderada, ou seja, calculando o domínio de cada neurônio.

Ao calcular os valores da solução, o domínio daquele neurônio passa a estar vinculado em toda a modelagem. Levando em conta todas as restrições já adicionadas, que pertencem às camadas anteriores, podem ser adicionadas as novas restrições relativas aos neurônios da modelagem. Essas restrições vão indicar o intervalo de valores (domínio) que o neurônio pode assumir, ou seja, o limite inferior e o superior calculados na solução vão auxiliar a definir essas restrições. Essas restrições equivalem à própria soma ponderada usada para a modelagem que acabamos de realizar, vide equação (2.1).

Generalizando, a soma ponderada do neurônio $x_j^{(k)}$ é definida por $\sum_{i=1}^n p_{i,k-1,j} x_i^{(k-1)} + v_j^{(k)}$, em que j e i são os índices dos neurônios em suas camadas, sendo i vinculado à camada $k - 1$ e j vinculado à camada k ; k é o índice da camada do neurônio; n é índice do último neurônio da camada $k - 1$. Já o domínio $D_j^{(k)}$ é representado por $[li_j^k, ls_j^k]$, em que li_j^k é o limite inferior do neurônio com índice j na camada k , que é encontrado através da solução de minimização de $x_j^{(k)}$, analogamente ls_j^k pode ser encontrado através da maximização de $x_j^{(k)}$.

Porém, para que a modelagem fique completa, precisamos incrementar o modelo com a função de ativação ReLU ao modelo MILP. Assim, a implementação requer algumas adaptações, uma vez que a ReLU tem comportamento não linear. Essas adaptações irão modificar o comportamento de saída dos neurônios, evidenciando a presença da função de ativação ReLU.

Dessa forma, uma variável de decisão $z_j^{(k)}$ pode ser adicionada às restrições, acarretando em uma representação objetiva da ReLU do neurônio $x_j(k)$. Essa variável de

decisão contribui para que o resultado do intervalo obtido do neurônio seja “forçado” a ser maior ou igual a zero. Se $z_j^{(k)}$ é igual a 0, a saída da ReLU é 0. Se $z_j^{(k)}$ é igual a 1, então a saída da ReLU é igual a parte linear da entrada e, assim, o comportamento da ReLU é corretamente simulado.

Assim sendo, devemos transformar a soma ponderada em uma restrição para ser computada pelo MILP, assim como a utilização da variável z_j^k , que associada ao limite inferior li_j^k e ao limite superior ls_j^k , deve ser adicionada em forma de restrições. A equação (3.1), adaptada de (TJENG; XIAO; TEDRAKE, 2017), define as restrições que modelam o comportamento de todos os neurônios da camada intermediária. As desigualdades introduzidas garantem que a variável binária controle a ativação do neurônio, permitindo sua ativação apenas quando a saída após a aplicação da ReLU for positiva. Além disso, essas restrições asseguram que a fórmula permaneça satisfatível mesmo quando a entrada do neurônio for negativa.

$$k = 1, \dots, K - 1 \left\{ \begin{array}{l} x_j^{(k)} \leq \sum_{i=1}^n p_{i,k-1,j} x_i^{(k-1)} + v_j^{(k)} - li_j^k (1 - z_j^k) \\ x_j^{(k)} \geq \sum_{i=1}^n p_{i,k-1,j} x_i^{(k-1)} + v_j^{(k)} \\ x_j^{(k)} \leq ls_j^k z_j^k \\ z_j^k \in \{0, 1\} \\ x_j^{(k)} \geq 0 \end{array} \right. \quad (3.1)$$

Desse modo, temos todas as características dos neurônios prontas para serem inseridas no modelo. Assim, o resultado de todos os neurônios da primeira camada intermediária serão os insumos da soma ponderada da próxima camada, e assim por diante. Isso posto, as iterações $k = 1, \dots, K - 1$ compõem o mesmo ciclo para todas as camadas intermediárias, na qual $K - 1$ representa a última camada intermediária.

Uma vez que na última camada não há incidência da função ReLU, os resultados dos neurônios da última camada intermediária irão alimentar o resultado dos neurônios $x_j^{(K)}$ da camada de saída K através da soma ponderada, sem a aplicação da ReLU. A equação (3.2) demonstra um exemplo da camada de saída em forma de restrições.

$$x_j^{(K)} = \sum_{i=1}^n p_{i,K-1,j} x_i^{(K-1)} + v_j^{(K)} \quad (3.2)$$

Assim sendo, detalhamos o passo a passo de como a RNA com incidência da ReLU pode ser representada por uma modelagem MILP. O resultado é um modelo que ainda pode ser modificado para diversas aplicações, como a de fornecer explicações, que iremos detalhar na seção 3.2.

3.2 Explicação Tradicional das RNAs

Dado que as RNAs possuem o comportamento de caixas pretas, existem inúmeros trabalhos que implementaram diferentes abordagens (ALGHOUL et al., 2018; MUSLEH et al., 2019) para tentar encontrar uma explicação para aquilo que as RNAs preveem. Neste trabalho, uma explicação pode ser entendida como conjunto de atributos da entrada que é suficiente para garantir a saída computada pela RNA, ou seja, a saída será a mesma para as entradas com os mesmos valores dos atributos no conjunto, independentemente de qualquer alteração dos atributos que não estão naquele conjunto.

Nesse trabalho utilizaremos a codificação de (TJENG; XIAO; TEDRAKE, 2017) com as atualizações de (CARVALHO; OLIVEIRA; ROCHA, 2024) para realizar a codificação e explicação de RNAs em MILP.

A abordagem de (IGNATIEV; NARODYTSKA; MARQUES-SILVA, 2019) resulta em uma chamada explicação minimal que elenca atributos encontrados pelo algoritmo responsáveis por determinar a classe de saída da RNA. Esta abordagem realiza o mapeamento com garantia formal de corretude por meio de formulações lógicas e raciocínio abdutivo.

Sendo assim, podemos definir uma fórmula F como o conjunto de restrições da equação (3.1) e da equação (3.2). Como já obtivemos o modelo que representa o comportamento preditivo da rede, representado por uma fórmula F , podemos tomar esse modelo para inserir as restrições que vão auxiliar a gerar as explicações. Por exemplo, considere uma rede neural artificial (RNA) com um conjunto de atributos de entrada dado por $E = \{x_1^{(0)}, x_2^{(0)}\}$, onde cada elemento representa uma variável real observável na entrada da rede. Suponha ainda que o conjunto de saída da rede seja $S = \{s_1, s_2\}$, em que cada $s_o \in S$ corresponde a uma classe possível do problema de classificação modelado pela RNA. Nesse contexto, a rede realiza o mapeamento dos atributos de entrada para uma das classes de saída, produzindo uma predição, assumindo uma instância na qual os valores de entrada dispostos em E resultam na saída s_1 .

Para facilitar esta demonstração, vamos considerar que todas as variáveis, pesos e vieses são números inteiros e que o domínio dos atributos de entrada só possam variar da seguinte forma: $D_1^{(0)} = [-5, +5]$ e $D_2^{(0)} = [-10, +8]$. Os valores $E = \{x_1^{(0)} = 5, x_2^{(0)} = 2\}$ resultam em uma saída $S = \{s_1 = 11, s_2 = -8\}$ e podem ser visualizados na Figura 10.

Com esse exemplo podemos notar que a saída s_1 é a previsão predominante para essa entrada E . No entanto, se alterarmos a saída e_1 para -1 , teremos uma nova previsão conforme a Figura 11.

Dessa forma, podemos perceber que a previsão predominante foi modificada para s_2 , quando a entrada e_1 foi alterada. Com isso, podemos considerar esse atributo como parte da explicação, pois algum valor do seu domínio modifica a configuração de saída de tal forma que altera a classificação prevista pela RNA.

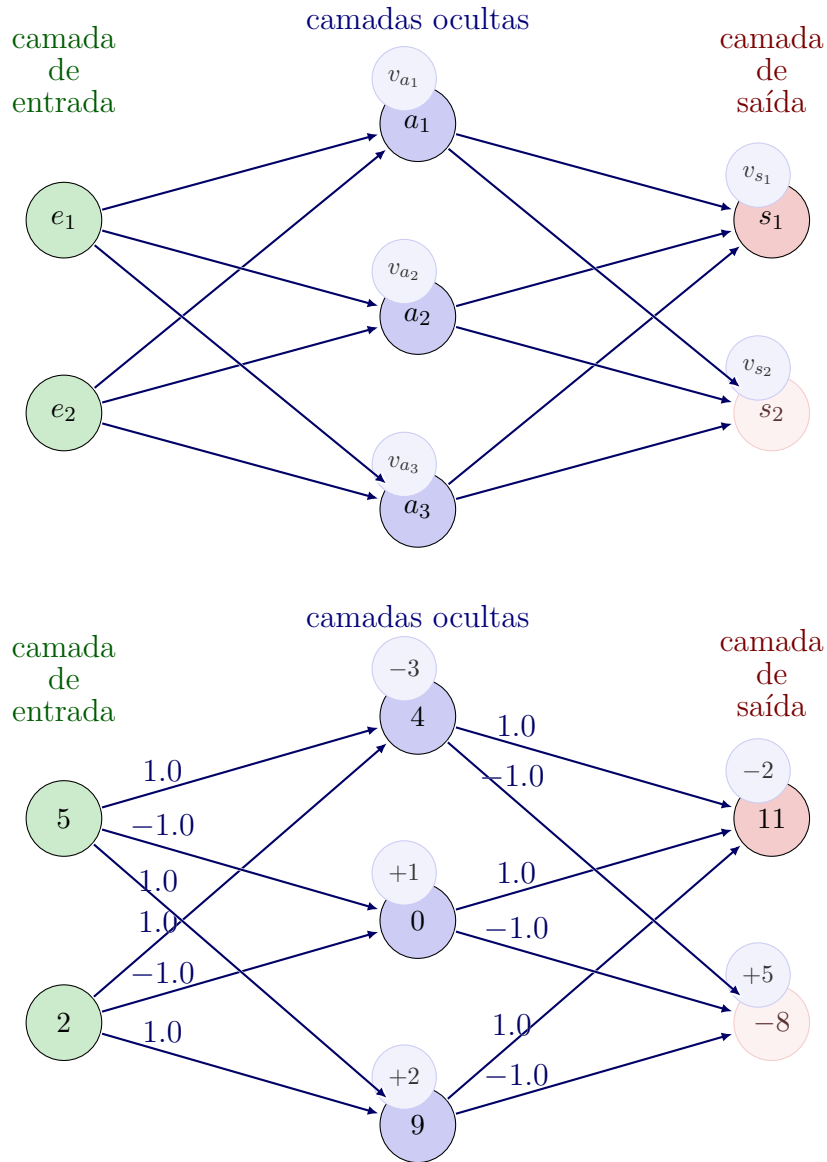


Figura 10 – Exemplo de previsão 1

Fonte: autoral

Da mesma forma, podemos tentar alterar o atributo e_2 com o valor do atributo e_1 permanecendo inalterado, conforme a Figura 12. Com algumas tentativas de alterar valores da entrada e_2 , vemos que isso não altera a previsão da saída do nosso exemplo. A função da modelagem com MILP é garantir que todos os valores possíveis dentro do domínio admitirão busca global (FAÍSCA et al., 2009) antes de considerar que um atributo da entrada não faz parte da explicação.

Desse modo, podemos inserir restrições nas variáveis de saída definindo qual saída deve ser predominante no modelo. No caso do nosso exemplo, a variável s_1 deve ser maior que s_2 e, caso houvesse outra variável de saída, aquela deveria ser maior que esta também. Nesse caso, com três variáveis de saída, teríamos como resultado a seguinte fórmula: $(s_1 > s_2) \wedge (s_1 > s_3)$.

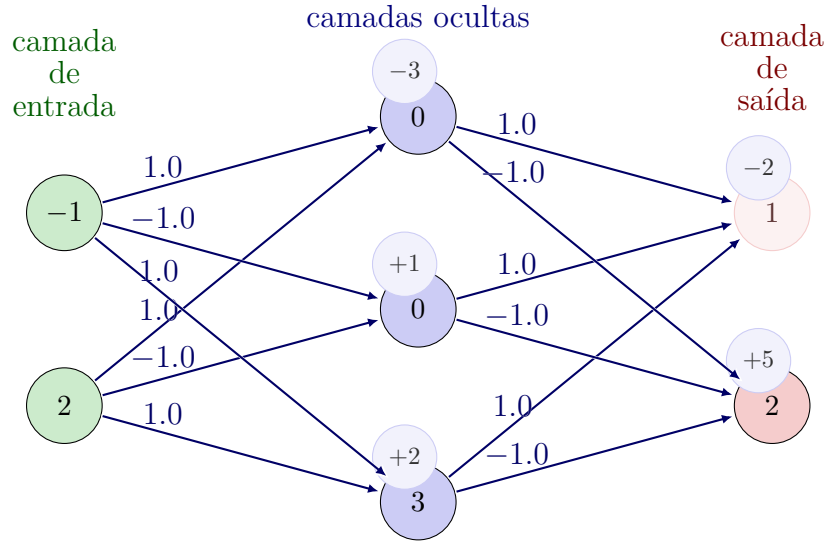


Figura 11 – Exemplo de previsão 2

Fonte: autoral

Generalizando, podemos obter um conjunto de conjunções lógicas conforme a equação (3.3), que representa a previsão da saída s_1 como a predição da RNA e pode ser adicionado à fórmula F . Para dar início à explicação, também é necessário definir um conjunto de restrições associado aos atributos da entrada no formato $E = \{x_1^{(0)} = v_1, x_2^{(0)} = v_2, \dots, x_k^{(0)} = v_k\}$.

$$S' = (s_1 > s_2) \wedge (s_1 > s_3) \wedge \dots \wedge (s_1 > s_n) \quad (3.3)$$

Dessa forma, a negação da equação (3.3) ($\neg S'$) representa que o valor da saída s_1 não é mais o maior. Assim, quando conseguimos demonstrar que a expressão $E' \wedge F \wedge \neg S'$ é insatisfatível, para $E' \subseteq E$, então o conjunto de restrições em E' são uma explicação dessa saída. Para verificar a insatisfabilidade dessa expressão, todos os valores possíveis dentro do domínio para os atributos de entrada que não estão no conjunto de entrada E' devem ser avaliados. A negação da equação (3.3) a transforma em um conjunto de disjunções lógicas com as relações de comparação invertidas, conforme a equação (3.4).

$$\neg S' = (s_1 \leq s_2) \vee (s_1 \leq s_3) \vee \dots \vee (s_1 \leq s_n) \quad (3.4)$$

Para representar essa equação $\neg S'$ no modelo MILP, podemos utilizar variáveis binárias q_i que representam as classes que não são a previsão e os limites do domínio de saída l_i e ls_i , com l sendo o índice dos neurônios da camada de saída K . Seja s_j representando a previsão, para cada i , a saída deverá respeitar a equação $s_j \leq s_i$, se q_i for maior ou igual a 1. Dessa forma, caso alguma dessas condições for satisfeita na qual q_i para algum $i \neq j$, $i \in \{1, \dots, n\}$ resulte $q_i = 1$, então $s_j \leq s_i$, deixando $\neg S'$ verdadeira.

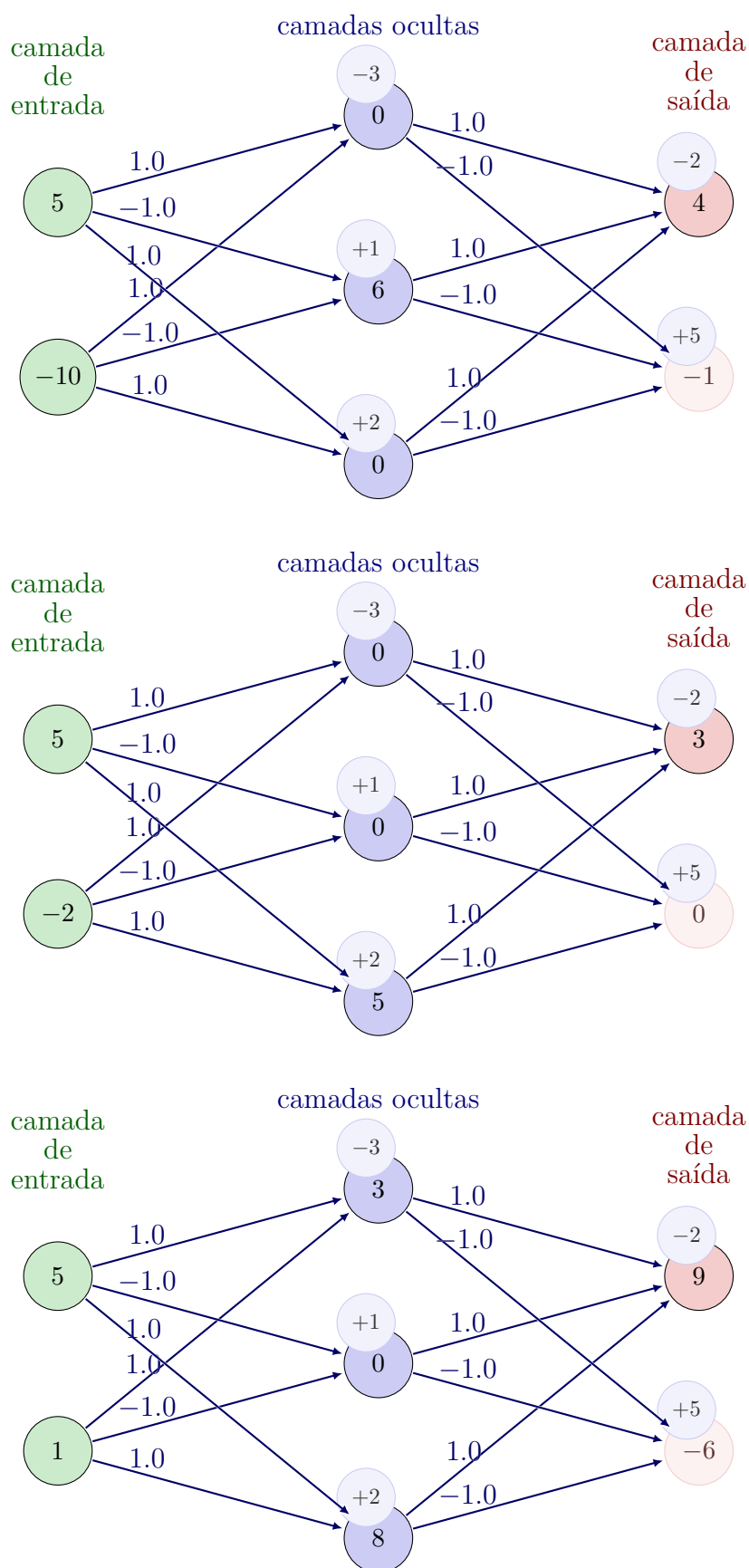


Figura 12 – Exemplo de previsão 3

Fonte: autoral

Senão, todo q_i será igual a zero, demonstrando a insatisfabilidade. Podemos visualizar essas restrições por meio da equação (3.5).

$$\begin{cases} s_j \leq s_i + (ls_j - ls_i)(1 - q_i), \\ \sum_{i=1}^k q_i \geq 1 \\ q_i \in \{0, 1\}, \forall i \in \{1, \dots, n\}, i \neq j \end{cases} \quad (3.5)$$

Após a definição das restrições da camada de saída, a geração da explicação segue o procedimento descrito por (IGNATIEV; NARODYTSKA; MARQUES-SILVA, 2019). O algoritmo consiste em iterar sobre cada atributo de entrada individualmente, removendo temporariamente sua restrição de valor a fim de permitir a avaliação de diferentes atribuições. Caso a remoção do atributo altere a saída do modelo, sua restrição original é reinserida, considerando-o como parte da explicação.

O Algoritmo 1 define os passos para a explicação com adaptações às nomenclaturas do exemplo citado mais acima.

Algoritmo 1 Explicação

Entrada: (Modelo F , Dados de Entrada E , previsão S)

Saída: Explicação mínima E_m

para todo $e = v$ em E **fazer**

$E' = E - \{e = v\}$

se $E' \wedge F \not\models S$ **então**

$E \leftarrow E' \cup \{e = v\}$

senão

$E \leftarrow E'$

fim se

fim para

$E_m \leftarrow E$

return E_m

Podemos notar que remover um atributo de entrada e com valor v transforma a entrada original E em uma nova entrada $E' = E - \{e = v\}$ podendo ser alterada conforme a explicação vai acontecendo no laço de repetição. O atributo e será sempre a representação de algum neurônio da camada de entrada $x_i^{(0)}$. Ao fim, o algoritmo retorna um subconjunto da entrada $E_m \subseteq E$ que é a explicação minimal do dado de entrada para a RNA.

4 Relaxamento das Variáveis Binárias para Fornecer Explicações

Conforme discutido na Seção 1.2, o uso de variáveis binárias impõe um custo computacional elevado aos modelos baseados em Programação Linear Inteira Mista. Diante disso, este capítulo aborda a técnica de relaxação dessas variáveis com a finalidade de viabilizar a computação de explicações de forma mais rápida. São discutidas, ainda, as implicações teóricas e práticas decorrentes dessa modificação, bem como suas interpretações nesse contexto. Por fim, detalha-se a aplicação da abordagem proposta, evidenciando os procedimentos adotados para maximizar a qualidade dos resultados obtidos.

4.1 Codificação Relaxada

A codificação relaxada preserva a estrutura geral de representação por meio de variáveis e restrições lineares. No entanto, distingue-se pela reformulação das variáveis de decisão, originalmente binárias, para variáveis contínuas.

As relaxações consideradas nessa abordagem, conforme discutido na Seção 2.2.1, consistem em permitir que as variáveis binárias assumam valores no intervalo contínuo $[0, 1]$. Tal modificação amplia o domínio de busca, transformando um problema originalmente discreto em contínuo. Especificamente, essa relaxação altera a equação (3.1), que impõe restrições sobre o domínio dos neurônios nas camadas intermediárias, impactando, assim, o domínio de todos os neurônios subsequentes à camada de entrada, inclusive os da camada de saída. Como resultado há a definição da equação (4.1).

$$k = 1, \dots, K - 1 \left\{ \begin{array}{l} x_j^{(k)} \leq \sum_{i=1}^n p_{i,k-1,j} x_i^{(k-1)} + v_j^{(k)} - l_j^k (1 - z_j^k) \\ x_j^{(k)} \geq \sum_{i=1}^n p_{i,k-1,j} x_i^{(k-1)} + v_j^{(k)} \\ x_j^{(k)} \leq l_j^k z_j^k \\ z_j^k \in [0, 1] \\ x_j^{(k)} \geq 0 \end{array} \right. \quad (4.1)$$

$$x_j^{(k)} = \sum_{i=1}^n p_{i,k-1,j} x_i^{(k-1)} + v_j^{(k)} \quad (4.2)$$

$$x_j^{(k)} \leq x_j'^{(k)} - (-5) \cdot (1 - z_j^k) \quad (4.3)$$

$$x_j^{(k)} \geq x_j'^{(k)} \quad (4.4)$$

$$x_j^{(k)} \leq 5 \cdot z_j^k \quad (4.5)$$

A modificação a ser notada é a reformulação das variáveis $z_j^k \in \{0, 1\}$ para $z_j^k \in [0, 1]$. Essa conversão não remove as propriedades preditivas da rede nem invalida a função de ativação ReLU, mas permite uma “ativação parcial” do neurônio mantendo as propriedades essenciais de retornar apenas valores não negativos (GRIMSTAD; ANDERSSON, 2019).

A alteração no domínio das variáveis decorre da relaxação, a qual pode modificar a solução ótima originalmente obtida. Essa modificação na solução ótima é especialmente observada em contextos nos quais se busca determinar os valores mínimo e máximo de ativação de um neurônio. Como resultado, a solução relaxada pode atribuir valores contínuos a variáveis que originalmente representariam decisões discretas, o que pode comprometer a interpretabilidade e a fidelidade lógica do modelo. A Seção 4.2.1 demonstra que essa modificação, embora remova a minimalidade das explicações geradas por meio da relaxação, não compromete a sua garantia de corretude.

Na Figura 13, apresenta-se um exemplo ilustrativo dos pesos associados aos neurônios de uma rede neural artificial (RNA) com função de ativação ReLU sendo aplicada em um nó distinto para melhor entendimento. A notação $x_2'^{(1)}$ representa a entrada do neurônio 2 da camada 1 antes de passar pela função de ativação ReLU, após isso, torna-se $x_2^{(1)}$. Esses pesos representam os coeficientes que poderiam ser extraídos ou calculados durante o processo de codificação da rede em um modelo formal. Tal exemplo será utilizado como base ao longo deste capítulo para demonstrar as etapas envolvidas na formulação das restrições, relaxações e explicações geradas a partir da análise estrutural da RNA.

4.2 Explicação Relaxada

A explicação relaxada produzirá um conjunto de atributos que garante a mesma saída do modelo sempre. O Algoritmo 1 pode ser reutilizado para produzir as explicações. Na verdade, não é necessária nenhuma alteração nessa etapa para produzir as explicações com garantia de corretude, no entanto não há como assegurar explicações minimais quando há relaxações no modelo.

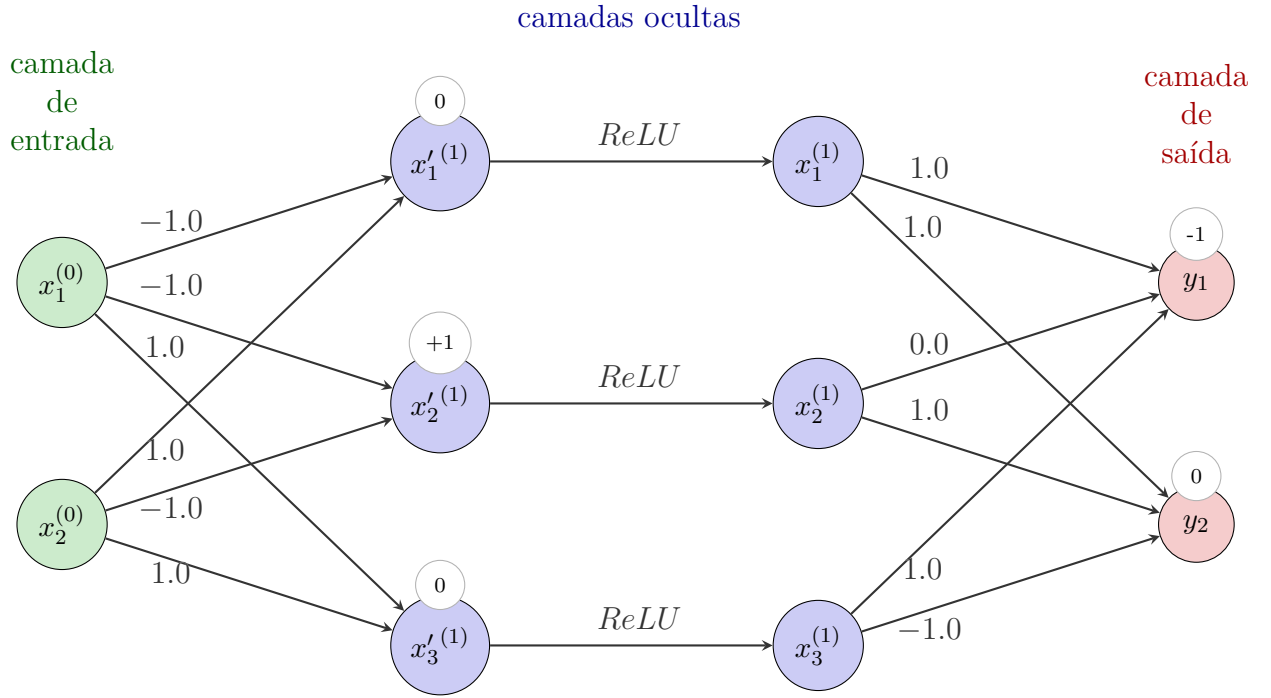


Figura 13 – Domínio dos neurônios de uma RNA com ReLU

Fonte: autoral

4.2.1 Interpretação dos Intervalos

Os intervalos obtidos para o domínio dos neurônios, decorrentes da relaxação, diferem daqueles utilizados na codificação original proposta por (CARVALHO; OLIVEIRA; ROCHA, 2024). Essa divergência levanta questionamentos sobre a natureza das explicações geradas. Como o passo da explicação no algoritmo visa satisfazer uma fórmula lógica que modifique a saída do modelo, a utilização de intervalos relaxados aumenta a probabilidade de que tais soluções sejam satisfeitas. Consequentemente, isso pode induzir a uma interpretação mais abrangente, atribuindo maior relevância a atributos que, sob a codificação original, poderiam não ser considerados essenciais para a decisão do modelo.

A Figura 14 ilustra como a relaxação de uma variável binária altera o intervalo de valores admissíveis para a variável $x_j^{(k)}$, a qual representa a entrada do neurônio, bem como os valores resultantes em sua saída correspondente $ReLU_{\text{relaxada}}(x_j^{(k)})$. A região sombreada indica os possíveis valores da saída do neurônio $x_j^{(k)}$. Esse processo de relaxação, quando aplicado a determinados neurônios, aproxima a modelagem original, intrinsecamente discreta, de um problema linear contínuo, cuja resolução possui complexidade polinomial. Ademais, observa-se que a modificação da saída do neurônio relaxado mantém uma propriedade fundamental da função ReLU, isto é, a não negatividade, como evidenciado pela região factível gerada pela relaxação.

Retomando a ideia da seção 3.2, seja F as equações (3.1) e (3.2), seja $E' \subseteq E$ um subconjunto da entrada E que vamos avaliar se é suficiente para garantir a predição,

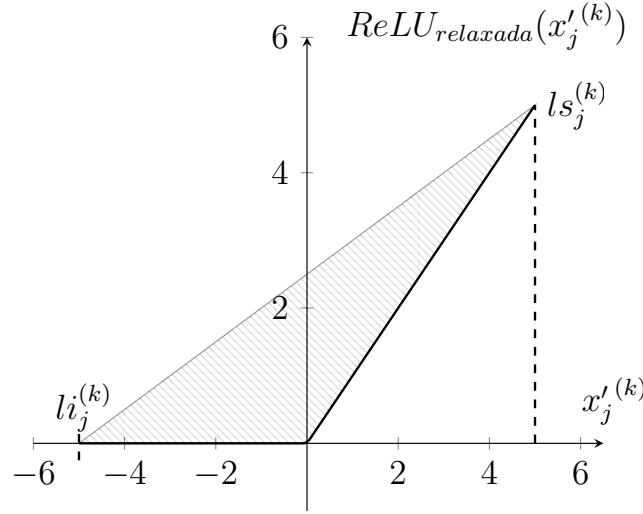


Figura 14 – Função da ReLU com relaxação

Fonte: autoral

seja S' conforme a equação (3.3), podemos demonstrar a satisfatibilidade da expressão $E' \wedge F \wedge S'$, sendo E' uma explicação minimal para a saída S' .

Considere o conjunto de restrições dos atributos de entrada no formato $E'_r = \{x_1^{(0)} = v_1, x_2^{(0)} = v_2, \dots, x_k^{(0)} = v_k\}$, a fórmula com restrições relaxadas F_r definida pelas equações (4.1) e (3.2), e a expressão S' representando as restrições da classificação atribuída pela RNA na equação (3.3). Podemos assumir que E'_r é uma explicação válida, uma vez que $E'_r = E$ e a expressão $E' \wedge F \wedge \neg S'$ é insatisfatível.

Embora essa verificação formal da satisfatibilidade não tenha, por si só, uma aplicação direta em problemas do mundo real, ela é essencial para fundamentar a validade lógica da explicação relaxada. Uma vez que, mesmo quando as variáveis binárias da formulação original são relaxadas, resultando potencialmente em saídas que não correspondem exatamente àsquelas produzidas pela rede neural original, nessa abordagem a relaxação permite verificar se certas alterações na entrada continuam levando a uma saída compatível com aquela originalmente classificada pela RNA.

Ainda que a relaxação amplie a região factível e produza uma saída que representa uma solução admissível no modelo contínuo, essa estrutura pode ser utilizada para analisar a robustez da decisão da rede. Se uma explicação relaxada preserva a satisfatibilidade da expressão $E'_r \wedge F_r \wedge S'$, então existe evidência formal de que essa entrada permanece dentro da região de decisão associada à saída S' . De modo geral, qualquer entrada $E'_r = E'$ pode ser verificada como explicação através da equação (4.6).

$$E'_r \wedge F_r \models S' \implies E' \wedge F \models S' \quad (4.6)$$

Dessa forma, se a equação 4.7 é insatisfatível, então a equação 4.8 também é insatisfatível. Consequentemente, a insatisfatibilidade da equação (4.7) demonstra que o conjunto de restrições E'_r são uma explicação para a saída classificada pela RNA.

$$E'_r \wedge F_r \wedge \neg S' \quad (4.7)$$

$$E' \wedge F \wedge \neg S' \quad (4.8)$$

Portanto, mesmo que o modelo relaxado possa, em alguns casos, não representar fielmente o comportamento exato da RNA devido à remoção da natureza discreta das variáveis binárias, ele ainda possui corretude do ponto de vista de explicação, especialmente para verificar propriedades como a estabilidade da classificação frente a perturbações de entrada. Na Figura 15 é possível visualizar uma saída produzida pela RNA da Figura 14 com o exemplo de entrada $E = \{x_1^{(0)} = 0.75, x_2^{(0)} = 0.75\}$.

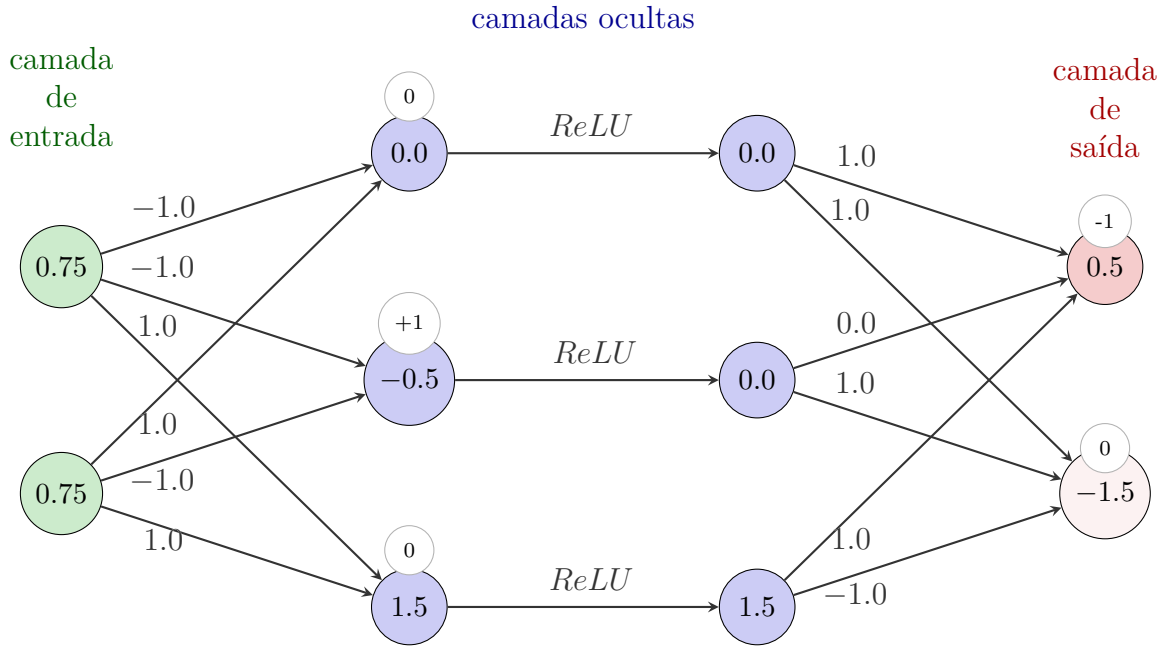


Figura 15 – Exemplo de saída sem relaxação

Fonte: autoral

A fórmula de cada neurônio da camada intermediária é satisfeita com valores para as variáveis binárias $Z = \{z_1^{(1)} = 1, z_2^{(1)} = 0, z_3^{(1)} = 1\}$. A saída para o neurônio $x_2^{(1)}$ é satisfeita pela equação (4.9) com valor de $x_2^{(1)}$ igual a 0. Os valores constantes encontrados para limite superior $ls_2^{(1)}$ e limite inferior $li_2^{(1)}$ são 1 e -1, respectivamente, e já foram substituídos diretamente nas restrições.

$$\begin{cases} x_2^{(1)} \leq -1 \cdot 0.75 - 1 \cdot 0.75 + 1 - (-1) \cdot (1 - z_2^{(1)}) \\ x_2^{(1)} \geq -1 \cdot 0.75 - 1 \cdot 0.75 + 1 \\ x_2^{(1)} \leq 1 \cdot z_2^{(1)} \\ z_2^{(1)} = 0 \\ x_2^{(1)} \geq 0 \end{cases} \quad (4.9)$$

Para demonstrar que $x_1^{(0)}$ não faz parte da explicação, é possível removê-lo do conjunto de entrada $E' = E \setminus \{x_1^{(0)} = 0.75\}$ e verificar que a classificação da RNA não é modificada. A Figura 16 ilustra um exemplo em que a variável $x_1^{(0)}$ assume o valor 0.0 sem que a saída da rede seja modificada, sugerindo que essa variável não é necessária para justificar a decisão da RNA. Portanto, caso não exista nenhum valor de $x_1^{(0)}$ cuja alteração impacte a saída da rede, essa variável pode ser descartada da explicação minimal. O modelo do exemplo na Figura 16 foi devidamente testado com as mesmas restrições e o solucionador não foi capaz de encontrar um valor para a variável $x_1^{(0)}$ que modificasse a classe.

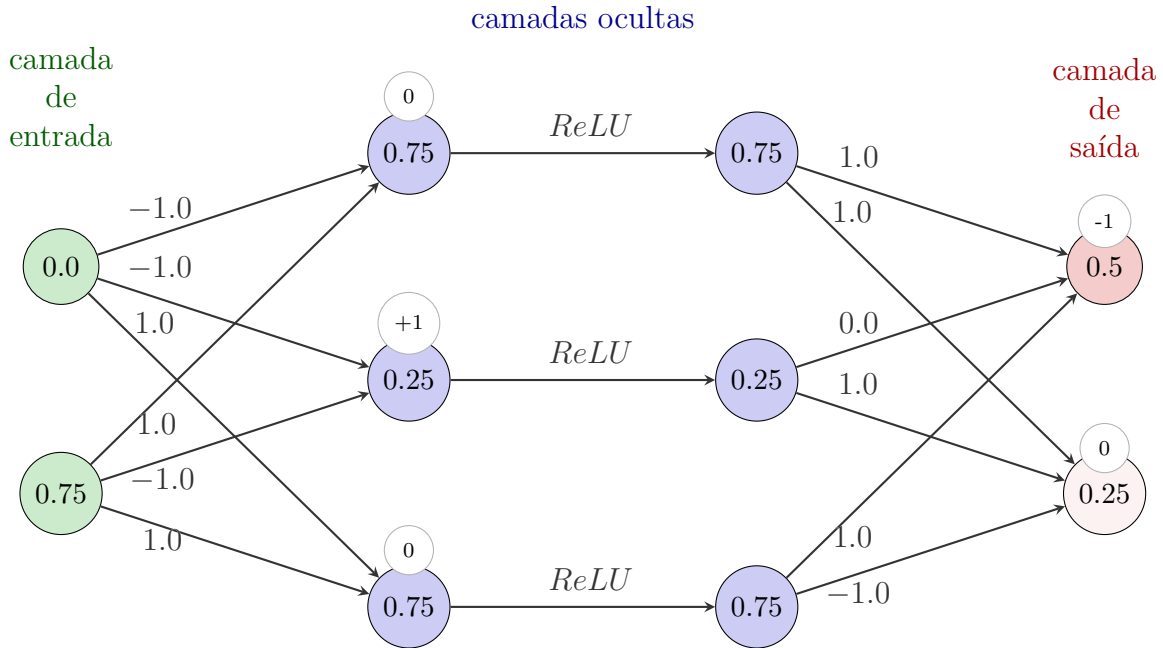


Figura 16 – Exemplo de saída sem relaxação que não modifica a classificação de saída

Fonte: autoral

Por outro lado, a Figura 17 mostra que, ao se aplicar uma relaxação na variável binária $z_2^{(1)}$, surge uma saída alternativa, ou seja, ocorre uma mudança na classificação. Isso indica que a ativação representada por $z_2^{(1)}$ influencia diretamente o resultado da RNA e, portanto, sendo considerada como parte da explicação nessa abordagem.

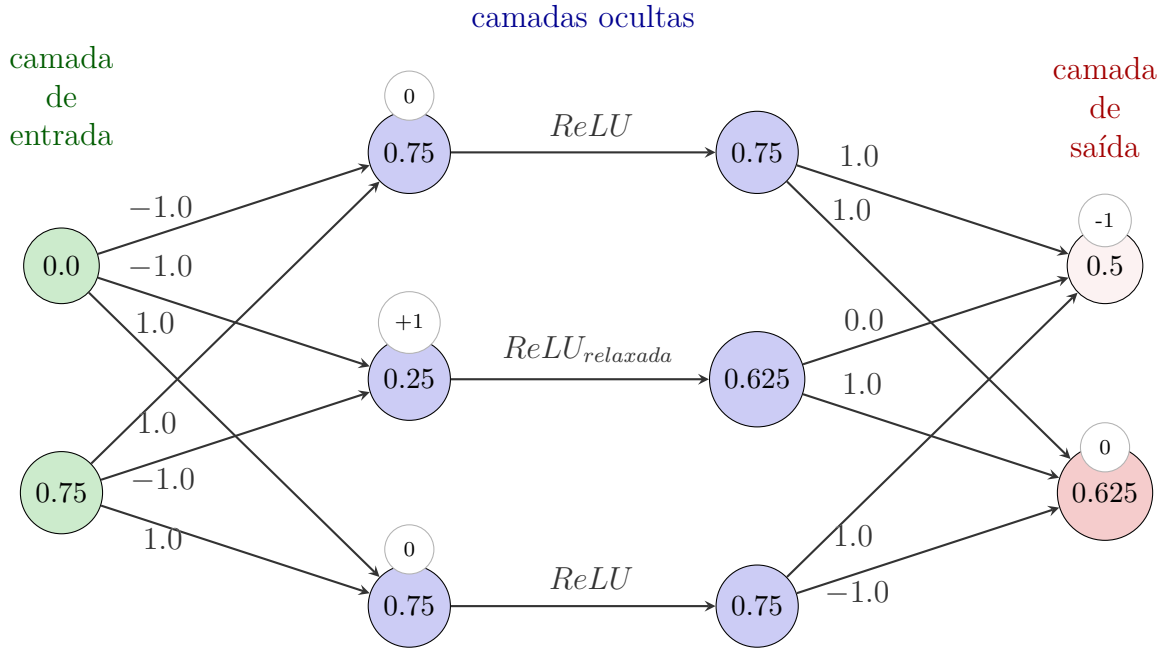


Figura 17 – Exemplo de saída com relaxação que modifica a classificação de saída

Fonte: autoral

$$\begin{cases} x_2^{(1)} \leq -1 \cdot 0.0 - 1 \cdot 0.75 + 1 - (-1) \cdot (1 - z_2^{(1)}) \\ x_2^{(1)} \geq -1 \cdot 0.0 - 1 \cdot 0.75 + 1 \\ x_2^{(1)} \leq 1 \cdot z_2^{(1)} \\ z_2^{(1)} = 0.625 \\ x_2^{(1)} \geq 0 \end{cases} \quad (4.10)$$

A equação (4.10) demonstra a saída do neurônio $x_2^{(1)}$ com restrições que satisfazem o valor presente na Figura 17. Finalmente, é possível notar que a relaxação de $z_2^{(1)} = 0.5$ não anula a garantia de corretude, uma vez que nenhum atributo foi desconsiderado como parte da explicação, mas deixou de identificar que poderia removê-lo.

Ainda no mesmo modelo, agora com uma modificação no domínio da entrada $x_1^{(0)}$ com seu limite inferior $li_1^{(0)}$ em 0.5, é possível notar que mesmo com a variável $z_2^{(1)}$ relaxada a classificação da saída não muda. A equação (4.11) sucede a equação (4.10) com valores que satisfaçam as novas restrições. Já a Figura 18 apresenta os valores obtidos através do solucionador para os todos os neurônios com essa mudança. Também foi constatado que nenhum outro valor foi capaz de modificar a classificação da saída mesmo com a aplicação da relaxação.

$$\begin{cases} x_2^{(1)} \leq -1 \cdot 0.5 - 1 \cdot 0.75 + 1 - (-0.5) \cdot (1 - z_2^{(1)}) \\ x_2^{(1)} \geq -1 \cdot 0.0 - 1 \cdot 0.75 + 1 \\ x_2^{(1)} \leq 0.5 \cdot z_2^{(1)} \\ z_2^{(1)} = 0.5 \\ x_2^{(1)} \geq 0 \end{cases} \quad (4.11)$$

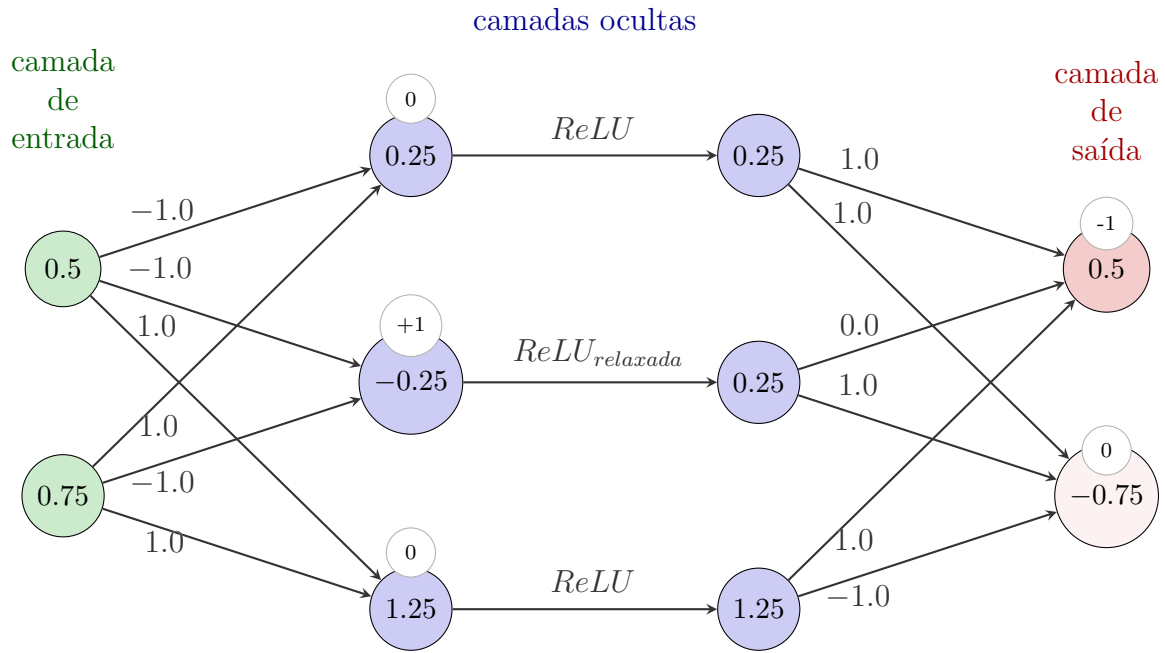


Figura 18 – Exemplo de saída com relaxação que não modifica a classificação de saída

Fonte: autoral

5 Experimentos

Este capítulo define os passos dos experimentos e seus respectivos resultados, detalhando as configurações, métodos e estruturas de dados utilizadas a fim de constatar os efeitos produzidos neste trabalho.

5.1 Ferramentas de Software

As arquiteturas consistem nas definições, em linguagem natural, da linguagem de programação, das bibliotecas, dos conceitos teóricos e do hardware utilizado para executar os experimentos. Foi utilizada a linguagem de programação Python¹ em toda a implementação, acessando algumas bibliotecas de código aberto para automatização dos testes e exposição dos dados. E paralelamente a biblioteca PyTorch (PASZKE et al., 2019) para treinar e manusear as redes neurais. Também foi utilizado o IBM-CPLEX² através da biblioteca DOcplex³, que está disponível para a linguagem Python, para modelar e solucionar problemas de Programação Linear Inteira Mista.

5.1.1 Hardware Utilizado

Todas as execuções referentes ao treinamento e validação das Redes Neurais Artificiais (RNAs), à modelagem dos programas de Programação Linear Inteira Mista (MILP) e à realização dos experimentos computacionais foram conduzidas na mesma máquina. O ambiente de experimentação consistiu em um sistema operacional Windows 11, executado sobre um hardware equipado com processador Intel Core i5-13450HX, 16 GB de memória RAM e uma GPU NVIDIA RTX 4050, sendo esta última utilizada exclusivamente para o treinamento das RNAs.

5.1.2 Repositórios dos Conjuntos de Dados

Foram utilizados 8 conjuntos de dados para condução dos experimentos desse trabalho. O repositório *UCI Machine Learning*⁴ foi usado como fonte para os seguintes conjuntos de dados:

¹ <<https://www.python.org/>>

² <<https://www.ibm.com/br-pt/analytics/cplex-optimizer>>

³ <<https://www.ibm.com/docs/en/icos/12.9.0?topic=docplex-python-modeling-api>>

⁴ Markelle Kelly, Rachel Longjohn, Kolby Nottingham. *The UCI Machine Learning Repository*. 2025. <<https://archive.ics.uci.edu/datasets/>>. Acesso em: 25 jul. 2025.

- **Wine:** conjunto de dados com 13 atributos contínuos, referentes a análises químicas de vinhos provenientes de três cultivares diferentes. Contém 178 instâncias e é amplamente utilizado para tarefas de classificação multiclasse.
- **Iris:** conjunto de dados com 4 atributos contínuos, correspondentes a medidas físicas de flores de três espécies do gênero *Iris* (setosa, versicolor e virginica). Contém 150 instâncias e é amplamente utilizado para tarefas de classificação multiclasse e validação de métodos de aprendizado de máquina.
- **Diabetes (Pima Indians Diabetes Database):** contém 8 atributos referentes a medições clínicas em mulheres da etnia Pima. O objetivo é prever a ocorrência de diabetes. São 768 instâncias no total.
- **Glass Identification:** conjunto com 9 atributos extraídos da análise química de fragmentos de vidro com o objetivo de classificar o tipo de vidro (ex: janelas, recipientes, faróis etc). Possui 214 instâncias e apresenta uma tarefa de classificação de 6 possibilidades.
- **Breast Cancer Wisconsin (Diagnostic):** conjunto com 30 atributos extraídos de imagens digitalizadas de células mamárias, utilizados para prever se um tumor é maligno ou benigno. Contém 569 instâncias. Também é amplamente adotado em benchmarks de classificação binária.
- **Sonar:** composto por 60 atributos que representam amplitudes de sinais de retorno de sonar. O objetivo é distinguir entre minas e rochas submarinas. Possui 208 instâncias e é um desafio clássico de classificação binária com alta dimensionalidade.

Além desses, outros dois conjuntos de alta dimensionalidade e complexidade para tarefas de classificação foram aproveitados:

- **Digits (Optical Recognition of Handwritten Digits):** conjunto de dados com 64 atributos correspondentes às intensidades de pixels de imagens 8x8 de dígitos manuscritos (dígitos de 0 a 9). Possui 1.797 instâncias e é usado para tarefas de classificação multiclasse com 10 classes. Está disponível no módulo `sklearn.datasets` do Scikit-learn (PEDREGOSA et al., 2011). A origem do conjunto está associada a um repositório experimental do UCI, mas sua versão mais amplamente utilizada atualmente está integrada ao Scikit-learn⁵ com modificações.
- **MNIST (Modified National Institute of Standards and Technology database):** um dos conjuntos de dados mais famosos da área de aprendizado de máquina, contém 70.000 imagens (60.000 para treino e 10.000 para teste) de dígitos manuscritos de 28x28 pixels (totalizando 784 atributos) (LECUN; CORTES; BURGESS, 1998). É usado para tarefas de classificação multiclasse com 10 clas-

⁵ PEDREGOSA, F. et al. *Scikit-learn: Machine Learning in Python*. 2011. 2825–2830 p. Disponível em: <<https://scikit-learn.org/>>.

ses (dígitos de 0 a 9). Disponível em diversos formatos e repositórios, incluindo o site oficial do Yann LeCun e bibliotecas como `tensorflow.keras.datasets` e `torchvision.datasets`.

5.1.3 Treinamento das Redes Neurais Artificiais

Para cada conjunto de dados, foram treinadas 6 Redes Neurais Artificiais padronizadas em relação a quantidade de neurônios por camada intermediária. As configurações de cada RNA são as seguintes:

- 2 camadas ocultas, 16 neurônios por camada;
- 2 camadas ocultas, 32 neurônios por camada;
- 3 camadas ocultas, 16 neurônios por camada;
- 3 camadas ocultas, 32 neurônios por camada;
- 4 camadas ocultas, 16 neurônios por camada;
- 4 camadas ocultas, 32 neurônios por camada;

Com o intuito de garantir o rigor científico compatível com o estado da arte na validação de redes neurais artificiais (RNAs), os treinamentos foram conduzidos de forma exaustiva, empregando otimizadores e hiperparâmetros selecionados por meio da técnica de Grid Search (LIASHCHYNSKYI; LIASHCHYNSKYI, 2019). Como resultado, foram obtidos modelos com acurácia superior a 90% nos conjuntos de dados MNIST, WINE, SONAR, IRIS e DIGITS. Para os conjuntos BREAST-CANCER, DIABETES e GLASS, os modelos atingiram acurácia superior a 79%, valor considerado satisfatório e compatível com os padrões observados na literatura para RNAs. Todos os detalhes da implementação do treinamento podem ser vistos no repositório público⁶.

Ressalta-se que não foi possível executar o experimento com RNAs a partir de 3 camadas com 32 neurônios para o conjunto MNIST, pois na versão comunitária disponível do solucionador CPLEX há um limite de 1000 variáveis e 1000 restrições para utilizar nos modelos de Programação Linear Inteira Mista. Os modelos citados ultrapassam esses valores no momento da explicação.

5.2 Condução

Para a condução dos experimentos, foram definidos quatro cenários distintos para a geração de explicações. Para cada RNA treinada, foram realizadas codificações com quatro diferentes níveis de relaxamento: sem relaxamento (0 relaxações), dois relaxamentos por camada oculta, quatro relaxamentos por camada oculta e oito relaxamentos por

⁶ FERNANDO, L. *Relaxing ReLU on ANNs for Explanations*. 2022. <<https://github.com/Luizfernandopq/XAI-nn-relu.git>>.

camada oculta. Isso resultou em um total de 36 codificações de RNAs para cada conjunto de dados analisado. Essas quantidades de variáveis binárias escolhidas para relaxamento foram definidas de forma heurística, com base na análise empírica da quantidade de atributos recorrentes nas explicações geradas ao longo do desenvolvimento experimental após tentativas com proporções variadas. Além disso a seleção de quais variáveis devem ser relaxadas acontece aleatoriamente dentro de cada camada.

A execução dos experimentos engloba explicações, utilizando cada uma das codificações. Nos conjuntos de dados com mais de 200 instâncias, foram sorteadas aleatoriamente 100 para gerar explicações. Nos conjuntos com quantidade inferior, as instâncias foram explicadas em sua totalidade.

Além disso, os resultados foram computados para cada codificação e seus parâmetros estão indicados a seguir:

- Tempo médio de explicação;
- Desvio Padrão do tempo de explicação;
- Tamanho médio de explicação;
- Desvio Padrão do tamanho de explicação;

A interpretação dos resultados será discutida na seção 5.3.

5.3 Resultados

Para fins comparativos, esta seção apresenta os resultados consolidados para todos os conjuntos de dados, organizados em tabelas que detalham o desempenho das explicações de cada RNA treinada em relação ao respectivo conjunto de dados na seção 5.3.1 para resultados gerais, e da seção 5.3.2 em diante há uma análise mais detalhada por conjunto.

5.3.1 Resultatos Gerais

Para pormenorizar melhor, os conjuntos de dados podem ser divididos em duas classes conforme a quantidade de atributos de entrada. A primeira classe reúne os conjuntos com baixa dimensionalidade, ou seja, com um número relativamente pequeno de atributos. Nela, incluem-se os conjuntos Diabetes, Glass, Iris e Wine que possuem respectivamente 8, 9, 4 e 13 atributos. Esses conjuntos costumam demandar menos recursos computacionais e são mais suscetíveis a técnicas exploratórias e visualizações diretas.

A segunda classe é composta por conjuntos de maior dimensionalidade, os quais trazem um número significativamente maior de atributos. É o caso dos conjuntos Breast Cancer, Digits, MNIST e Sonar com 30, 64, 60 e 784 atributos, respectivamente. Esses conjuntos requerem, com maior frequência, abordagens específicas para tratamento da

dimensionalidade, como normalização, redução de dimensionalidade, ou regularização, a fim de evitar problemas como overfitting e complexidade excessiva dos modelos.

A Tabela 1 expõe o tempo médio de explicação para cada quantidade de relaxações e configurações de camadas ocultas das RNAs dos conjuntos de dados com menos atributos. É possível notar uma melhor estabilidade no desvio padrão do tempo em configurações com muitos neurônios nas camadas ocultas, principalmente em casos que a relaxação foi aplicada em maior quantidade. O tempo médio necessário para gerar uma explicação tende a diminuir à medida que se aplicam mais relaxações ao modelo, seguindo uma relação aproximadamente inversa entre o número de relaxações e o tempo de processamento, com algumas exceções.

Conjunto	Camadas	0 relaxações	2 relaxações	4 relaxações	8 relaxações
Diabetes (8)	2x16	0.19 ± 0.05	0.15 ± 0.07	0.14 ± 0.06	0.11 ± 0.05
	2x32	0.34 ± 0.08	0.31 ± 0.11	0.28 ± 0.10	0.23 ± 0.08
	3x16	0.29 ± 0.05	0.25 ± 0.07	0.24 ± 0.08	0.20 ± 0.05
	3x32	0.50 ± 0.10	0.50 ± 0.12	0.43 ± 0.13	0.41 ± 0.11
	4x16	0.34 ± 0.06	0.28 ± 0.08	0.26 ± 0.06	0.17 ± 0.06
	4x32	0.74 ± 0.16	0.66 ± 0.17	0.62 ± 0.20	0.45 ± 0.12
Glass (9)	2x16	0.23 ± 0.05	0.24 ± 0.05	0.13 ± 0.05	0.12 ± 0.05
	2x32	0.39 ± 0.10	0.32 ± 0.11	0.27 ± 0.08	0.36 ± 0.11
	3x16	0.32 ± 0.08	0.26 ± 0.10	0.26 ± 0.11	0.16 ± 0.05
	3x32	1.71 ± 1.15	0.70 ± 0.21	0.50 ± 0.16	0.49 ± 0.14
	4x16	0.60 ± 0.19	0.31 ± 0.08	0.28 ± 0.07	0.23 ± 0.05
	4x32	4.08 ± 2.32	2.59 ± 8.69	1.77 ± 1.42	0.45 ± 0.11
Iris (4)	2x16	0.09 ± 0.03	0.09 ± 0.04	0.09 ± 0.04	0.07 ± 0.04
	2x32	0.12 ± 0.04	0.12 ± 0.04	0.12 ± 0.04	0.08 ± 0.04
	3x16	0.13 ± 0.04	0.12 ± 0.04	0.12 ± 0.04	0.10 ± 0.04
	3x32	0.25 ± 0.06	0.23 ± 0.07	0.22 ± 0.06	0.22 ± 0.07
	4x16	0.13 ± 0.04	0.12 ± 0.05	0.12 ± 0.06	0.11 ± 0.04
	4x32	0.38 ± 0.10	0.32 ± 0.05	0.30 ± 0.07	0.26 ± 0.06
Wine (13)	2x16	0.49 ± 0.10	0.47 ± 0.12	0.47 ± 0.13	0.37 ± 0.14
	2x32	0.67 ± 0.11	0.70 ± 0.11	0.65 ± 0.14	0.61 ± 0.16
	3x16	0.65 ± 0.11	0.65 ± 0.15	0.41 ± 0.16	0.36 ± 0.13
	3x32	1.09 ± 0.17	1.11 ± 0.17	1.06 ± 0.62	0.94 ± 0.27
	4x16	0.66 ± 0.14	0.61 ± 0.13	0.50 ± 0.13	0.23 ± 0.08
	4x32	2.39 ± 1.46	2.08 ± 0.87	1.87 ± 0.97	1.17 ± 0.59

Tabela 1 – Tempo Médio de Explicação ($\pm$ desvio padrão) por configuração dos conjuntos com menos de 30 atributos.

A Tabela 2 expõe o tamanho médio das explicações nos mesmos moldes da Tabela 1,

a fim de observar o impacto da relaxação na qualidade das explicações. É possível perceber que a quantidade de atributos removidos durante as explicações torna-se praticamente inexpressivo quando aplicamos muitas relaxações por camada.

Conjunto	Camadas	0 relaxações	2 relaxações	4 relaxações	8 relaxações
Diabetes (8)	2x16	5.69 ± 1.21	6.93 ± 1.23	6.43 ± 1.60	7.37 ± 1.15
	2x32	5.44 ± 1.28	6.26 ± 1.50	6.32 ± 1.55	7.39 ± 1.06
	3x16	4.78 ± 1.43	6.02 ± 1.51	6.96 ± 1.41	7.93 ± 0.35
	3x32	5.50 ± 1.29	5.95 ± 1.24	6.70 ± 1.35	7.14 ± 1.14
	4x16	5.53 ± 1.20	6.94 ± 1.35	7.31 ± 1.11	7.98 ± 0.14
	4x32	5.33 ± 1.44	6.17 ± 1.61	6.35 ± 1.75	7.78 ± 0.70
Glass (9)	2x16	6.92 ± 1.05	6.17 ± 1.02	8.95 ± 0.26	8.78 ± 0.63
	2x32	7.11 ± 1.19	8.21 ± 1.13	8.60 ± 0.82	6.24 ± 2.43
	3x16	7.44 ± 0.92	8.01 ± 1.11	7.62 ± 1.48	9.00 ± 0.00
	3x32	5.55 ± 1.39	7.53 ± 1.26	8.80 ± 0.55	8.81 ± 0.50
	4x16	6.08 ± 0.90	8.99 ± 0.10	8.96 ± 0.24	9.00 ± 0.00
	4x32	7.12 ± 0.82	8.51 ± 0.87	8.30 ± 1.02	9.00 ± 0.00
Iris (4)	2x16	2.48 ± 0.65	2.85 ± 0.80	2.88 ± 0.83	3.11 ± 0.83
	2x32	3.07 ± 0.54	3.21 ± 0.51	3.26 ± 0.48	3.72 ± 0.45
	3x16	2.89 ± 0.69	3.11 ± 0.82	3.16 ± 0.85	3.69 ± 0.52
	3x32	2.61 ± 0.67	3.23 ± 0.81	3.29 ± 0.66	3.44 ± 0.69
	4x16	2.52 ± 0.66	2.86 ± 0.79	3.10 ± 0.85	3.30 ± 0.85
	4x32	2.79 ± 0.60	3.01 ± 0.76	3.38 ± 0.51	3.53 ± 0.57
Wine (13)	2x16	9.40 ± 1.40	10.22 ± 1.34	10.99 ± 1.60	12.26 ± 1.25
	2x32	9.37 ± 1.17	9.79 ± 1.28	10.56 ± 1.59	11.27 ± 1.63
	3x16	9.44 ± 1.17	9.75 ± 1.33	11.67 ± 1.52	12.95 ± 0.34
	3x32	9.10 ± 1.26	9.30 ± 1.22	10.25 ± 1.49	10.97 ± 1.57
	4x16	9.48 ± 1.18	10.61 ± 1.29	11.96 ± 1.27	12.99 ± 0.15
	4x32	8.99 ± 1.37	9.52 ± 1.45	10.58 ± 1.59	11.90 ± 1.32

Tabela 2 – Tamanho Médio de Explicação ($\pm$ desvio padrão) por configuração dos conjuntos com menos de 30 atributos.

Por outro lado, a Tabela 3 expõe o tempo médio de explicação para cada quantidade de relaxações e configurações de camadas ocultas das RNAs dos conjuntos de dados com mais atributos. É possível notar uma redução substancial no desvio padrão do tempo em configurações com muitos neurônios nas camadas ocultas, principalmente em casos que a relaxação foi aplicada em maior quantidade.

Observa-se, na maioria dos casos, uma redução significativa no tempo médio de geração das explicações, especialmente nos conjuntos com maior número de atributos, o que corrobora a hipótese de que as relaxações podem tornar o processo computacionalmente

mais eficiente. A Tabela 3 destaca em **negrito** o caso que melhor equilibra a diminuição do tempo de explicação com o aumento no tamanho das explicações resultante das relaxações aplicadas.

Conjunto	Camadas	0 relaxações	2 relaxações	4 relaxações	8 relaxações
Breast Cancer (30)	2x16	1.08 ± 0.22	0.96 ± 0.27	0.83 ± 0.32	0.49 ± 0.18
	2x32	1.45 ± 0.27	1.45 ± 0.29	1.17 ± 0.42	0.84 ± 0.41
	3x16	1.04 ± 0.16	0.80 ± 0.21	0.75 ± 0.22	0.64 ± 0.14
	3x32	7.00 ± 10.65	7.47 ± 11.86	1.54 ± 0.63	1.01 ± 0.33
	4x16	1.32 ± 0.29	1.19 ± 0.39	0.75 ± 0.21	0.63 ± 0.14
	4x32	40.64 ± 127.24	8.73 ± 14.32	38.46 ± 101.83	1.29 ± 0.78
Digits (64)	2x16	2.98 ± 1.07	1.79 ± 1.00	1.36 ± 0.82	0.97 ± 0.28
	2x32	11.04 ± 29.45	3.94 ± 2.30	1.40 ± 1.30	1.16 ± 0.55
	3x16	4.36 ± 1.74	1.62 ± 1.02	1.26 ± 0.61	0.95 ± 0.24
	3x32	39.03 ± 109.97	18.23 ± 56.76	3.87 ± 6.73	1.38 ± 0.58
	4x16	77.23 ± 72.89	2.30 ± 7.19	1.18 ± 0.64	0.92 ± 0.20
	4x32	95.98 ± 174.86	23.63 ± 42.51	9.45 ± 6.97	4.38 ± 4.91
MNIST (784)	2x16	15.12 ± 3.68	11.15 ± 1.90	11.39 ± 2.53	10.71 ± 2.97
	2x32	54.72 ± 124.49	29.44 ± 21.09	24.80 ± 16.12	21.94 ± 8.71
	3x16	17.54 ± 3.79	14.78 ± 5.25	17.24 ± 6.70	12.36 ± 3.13
	4x16	24.65 ± 15.11	23.45 ± 10.33	20.54 ± 8.93	18.82 ± 8.63
Sonar (60)	2x16	1.96 ± 0.36	1.35 ± 0.61	0.85 ± 0.24	0.70 ± 0.14
	2x32	2.18 ± 0.49	1.47 ± 0.60	1.66 ± 0.62	0.93 ± 0.20
	3x16	2.17 ± 0.45	1.53 ± 0.58	1.11 ± 0.19	0.96 ± 0.17
	3x32	3.77 ± 0.83	3.10 ± 1.44	2.13 ± 1.46	1.16 ± 0.31
	4x16	1.73 ± 0.34	1.12 ± 0.46	0.87 ± 0.18	0.78 ± 0.13
	4x32	8.92 ± 8.52	7.34 ± 12.62	4.91 ± 7.45	2.11 ± 0.45

Tabela 3 – Tempo Médio de Explicação ($\pm$ desvio padrão) por configuração dos conjuntos com pelo menos 30 atributos. Em **negrito**, as configurações que lograram maior redução de tempo para o conjunto MNIST.

A Tabela 4 expõe o tamanho médio das explicações nos mesmos moldes da Tabela 3, a fim de observar o impacto da relaxação na qualidade das explicações. É possível perceber que a quantidade de atributos removidos durante as explicações torna-se praticamente inexpressivo quando aplicamos muitas relaxações por camada. Contudo, o conjunto MNIST constitui uma exceção a essa tendência. Em média, não se observa um aumento expressivo no tamanho das explicações, que permanecem aproximadamente 27% inferiores ao total de atributos disponíveis. Esse comportamento pode estar relacionado à natureza específica do problema tratado por esse conjunto de dados, sugerindo que uma quantidade maior de atributos pode viabilizar quantidades maiores de relaxação.

Conjunto	Camadas	0 relaxações	2 relaxações	4 relaxações	8 relaxações
Breast Cancer (30)	2x16	21.22 $\pm$ 3.02	23.43 $\pm$ 3.74	25.36 $\pm$ 4.04	29.43 $\pm$ 1.71
	2x32	20.32 $\pm$ 3.34	20.95 $\pm$ 3.50	25.15 $\pm$ 4.49	27.82 $\pm$ 4.13
	3x16	21.67 $\pm$ 3.70	25.83 $\pm$ 4.26	28.55 $\pm$ 2.97	29.97 $\pm$ 0.28
	3x32	22.25 $\pm$ 3.73	24.29 $\pm$ 4.36	26.77 $\pm$ 4.04	29.87 $\pm$ 0.34
	4x16	21.82 $\pm$ 2.85	25.55 $\pm$ 3.77	29.27 $\pm$ 2.26	30.00 $\pm$ 0.00
	4x32	21.37 $\pm$ 3.13	24.87 $\pm$ 4.32	24.54 $\pm$ 4.45	29.64 $\pm$ 1.94
Digits (64)	2x16	39.36 $\pm$ 9.48	47.25 $\pm$ 12.82	60.10 $\pm$ 7.03	64.00 $\pm$ 0.00
	2x32	42.42 $\pm$ 3.76	49.09 $\pm$ 8.62	60.46 $\pm$ 7.17	60.43 $\pm$ 6.40
	3x16	41.50 $\pm$ 6.51	51.99 $\pm$ 8.78	54.11 $\pm$ 8.16	63.14 $\pm$ 3.68
	3x32	43.78 $\pm$ 4.78	51.22 $\pm$ 9.23	60.45 $\pm$ 7.49	63.80 $\pm$ 1.54
	4x16	44.39 $\pm$ 3.33	63.19 $\pm$ 3.56	63.33 $\pm$ 2.78	63.51 $\pm$ 1.47
	4x32	38.21 $\pm$ 6.74	47.75 $\pm$ 10.14	48.51 $\pm$ 9.52	56.87 $\pm$ 10.39
MNIST (784)	2x16	615.20 $\pm$ 43.35	635.22 $\pm$ 61.81	624.57 $\pm$ 115.11	650.98 $\pm$ 119.25
	2x32	544.25 $\pm$ 52.96	574.82 $\pm$ 76.81	577.36 $\pm$ 103.60	600.42 $\pm$ 113.42
	3x16	569.93 $\pm$ 42.90	623.82 $\pm$ 80.22	572.53 $\pm$ 136.51	597.55 $\pm$ 109.12
	4x16	558.84 $\pm$ 49.67	550.34 $\pm$ 116.88	556.46 $\pm$ 114.06	572.80 $\pm$ 156.84
Sonar (60)	2x16	51.91 $\pm$ 3.07	55.85 $\pm$ 4.51	59.10 $\pm$ 2.33	59.98 $\pm$ 0.28
	2x32	52.00 $\pm$ 3.16	56.07 $\pm$ 4.41	54.92 $\pm$ 4.33	59.89 $\pm$ 0.75
	3x16	52.44 $\pm$ 3.04	58.33 $\pm$ 2.98	60.00 $\pm$ 0.00	60.00 $\pm$ 0.00
	3x32	52.22 $\pm$ 3.14	54.80 $\pm$ 4.04	58.04 $\pm$ 3.23	59.98 $\pm$ 0.14
	4x16	51.92 $\pm$ 3.00	57.82 $\pm$ 3.58	60.00 $\pm$ 0.00	60.00 $\pm$ 0.00
	4x32	52.50 $\pm$ 3.05	57.17 $\pm$ 4.00	58.21 $\pm$ 3.38	59.96 $\pm$ 0.27

Tabela 4 – Tamanho Médio de Explicação ($\pm$ desvio padrão) por configuração dos conjuntos com pelo menos 30 atributos.

Um aspecto relevante observado nos resultados é que a aplicação da relaxação contribui para a redução do tempo necessário na geração das explicações em todos os cenários avaliados. Além disso, constata-se que, nos conjuntos de dados com maior número de atributos, o uso da relaxação apresenta baixo impacto no aumento do tamanho das explicações, indicando que a técnica mantém a consistência da representação explicativa mesmo sob uma dimensionalidade elevada. Por exemplo, na configuração realçada em negrito da Tabela 3 foi constatada uma redução de até 59.9% no tempo médio de explicação para um aumento de apenas 10.32% no tamanho médio de explicação.

As mudanças mais significativas e notórias estão presentes nos modelos com maior quantidade de neurônios no total, que são justamente os modelos que fazem referência ao problema de escalabilidade.

As próximas seções exploram os resultados de forma mais detalhada para cada conjunto de dados, analisando o impacto da técnica de relaxação em relação ao tempo

de explicação e a complexidade da explicação gerada. Essa análise individual permite compreender melhor como as características de cada conjunto, como o número de atributos e quantidade de classes, podem tirar proveito da relaxação.

5.3.2 Conjunto Diabetes

Este conjunto (Pima Indians Diabetes) possui 768 instâncias e 8 atributos numéricos. Ele busca prever a ocorrência de diabetes em mulheres com mais de 21 anos, a partir de variáveis clínicas. Na Figura 19 é possível visualizar o tempo médio de explicação com relação a diferentes relaxamentos e configurações de camadas ocultas. O gráfico apresenta uma redução regular no tempo necessário para gerar explicações.

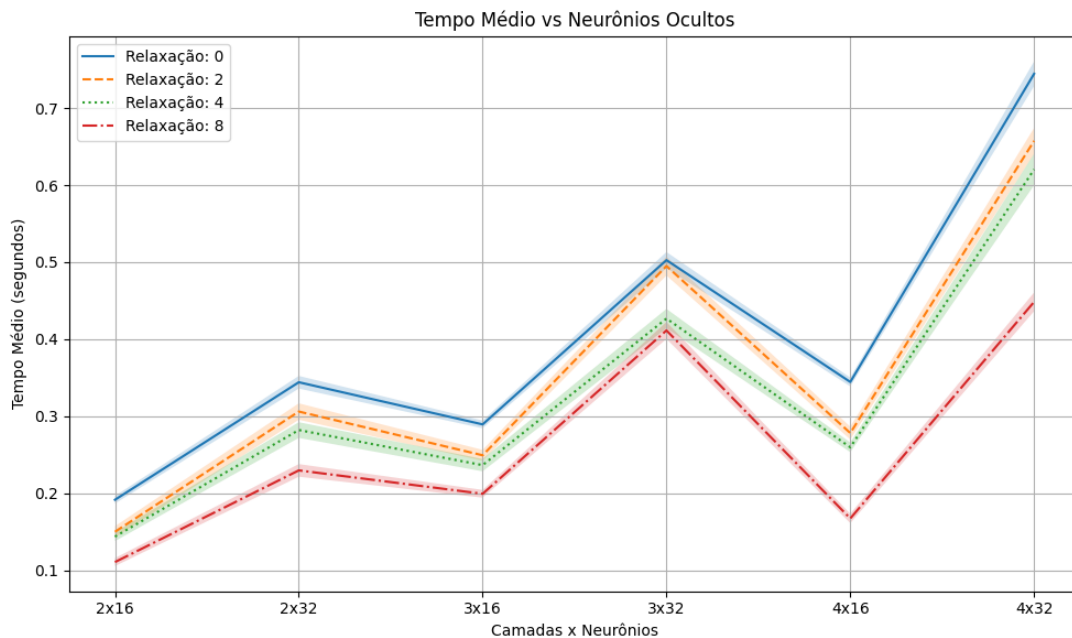


Figura 19 – Gráfico do Tempo Médio de Explicação para o conjunto de dados Diabetes, utilizando diferentes configurações de relaxação aplicadas às redes neurais.

Fonte: autoral

Na Figura 20 é possível visualizar o tamanho médio de explicação com relação a diferentes relaxamentos e configurações de camadas ocultas. O gráfico apresenta um aumento irregular no tempo necessário para gerar explicações, mas proporcional a quantidade de relaxações realizadas.

A variabilidade do tamanho das explicações com relação a diferentes configurações de RNAs se justifica pela precisão dos modelos e pelas instâncias sorteadas em cada execução. Para confirmar a validade, o sorteio foi realizado apenas entre configurações de RNAs, não relaxações.

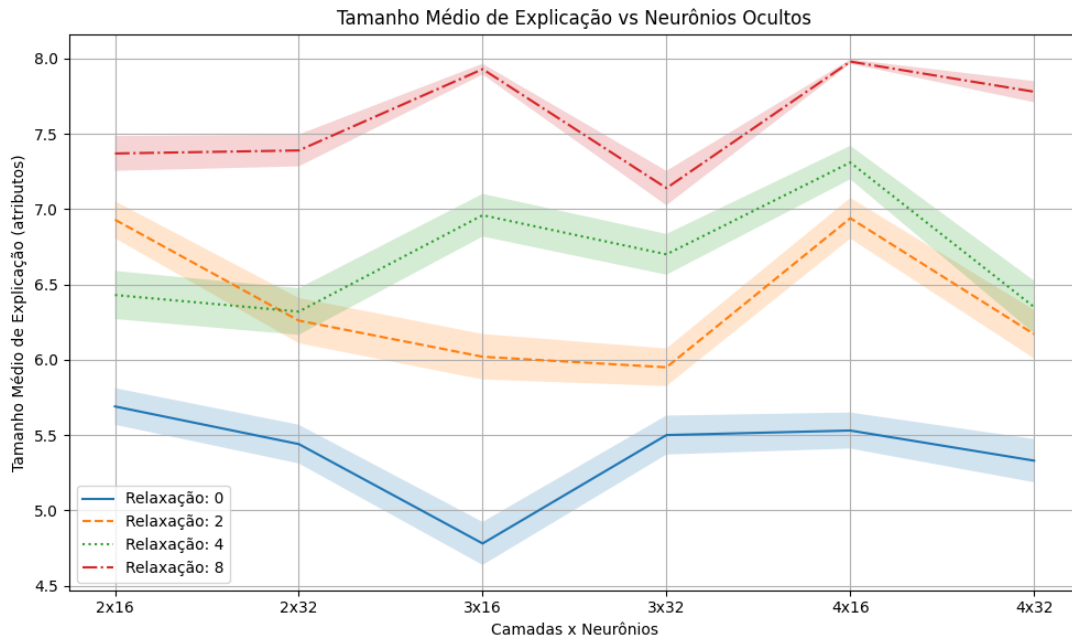


Figura 20 – Gráfico do Tamanho Médio de Explicação para o conjunto de dados Diabetes, utilizando diferentes configurações de relaxação aplicadas às redes neurais.

Fonte: autoral

5.3.3 Conjunto Glass

O conjunto Glass possui 214 instâncias e 9 atributos contínuos. Ele tem como objetivo identificar o tipo de vidro com base em sua composição química. Apesar da baixa dimensionalidade, a presença de seis classes e o número reduzido de amostras tornam o problema mais complexo. A multiplicidade de classes, em particular, destaca-se como um fator que contribui para o aumento do tempo necessário para a geração das explicações.

Na Figura 21 é possível visualizar o tempo médio de explicação com relação a diferentes relaxamentos e configurações de camadas ocultas. O gráfico apresenta uma redução mais significativa no tempo necessário para gerar explicações em Redes com mais neurônios nas camadas ocultas.

Na Figura 22, observa-se o tamanho médio das explicações em função dos diferentes níveis de relaxamento e configurações de camadas ocultas. Nota-se um aumento não linear, próximo ao número máximo de atributos do conjunto de dados, especialmente em redes com maior número de neurônios nas camadas ocultas.

A possibilidade de gerar explicações com tamanho menor em codificações relaxadas se justifica pela mudança na captura de quais atributos são considerados como parte da explicação, resultando em uma combinação diferente. Essa variação será mais bem explorada na seção 5.3.9.

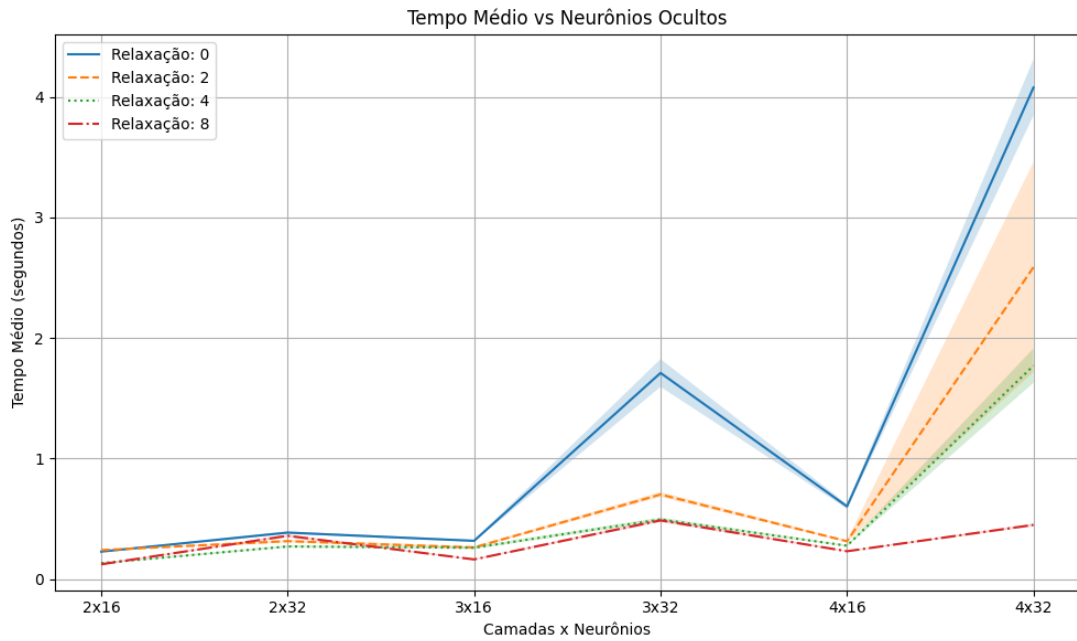


Figura 21 – Gráfico do Tempo Médio de Explicação para o conjunto de dados Glass, utilizando diferentes configurações de relaxação aplicadas às redes neurais.

Fonte: autoral

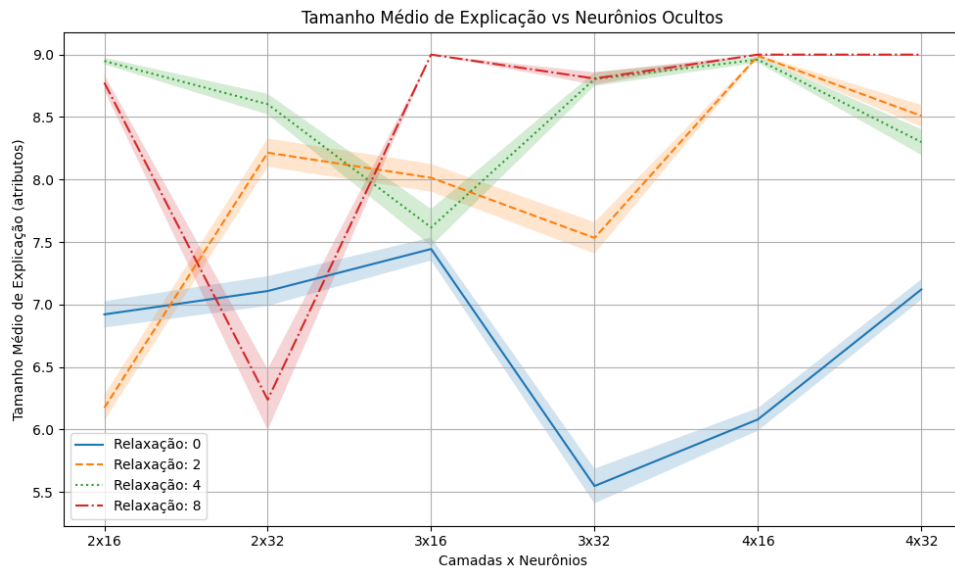


Figura 22 – Gráfico do Tamanho Médio de Explicação para o conjunto de dados Glass, utilizando diferentes configurações de relaxação aplicadas às redes neurais.

Fonte: autoral

5.3.4 Iris

O conjunto possui 150 amostras, distribuídas uniformemente entre três classes, totalizando 50 exemplos por classe. Cada amostra é representada por quatro atributos contínuos: comprimento e largura das sépalas, e comprimento e largura das pétalas.

Embora seja um conjunto de baixa dimensionalidade, sua simetria entre classes e a

separabilidade de duas delas (setosa e virginica) o tornam útil para avaliar rapidamente o desempenho e o comportamento dos algoritmos de explicação. Por apresentar um número reduzido de atributos e classes, as explicações tendem a ser mais rápidas de serem geradas, não sendo possível notar grandes reduções de tempo médio de explicação, conforme a Figura 23.

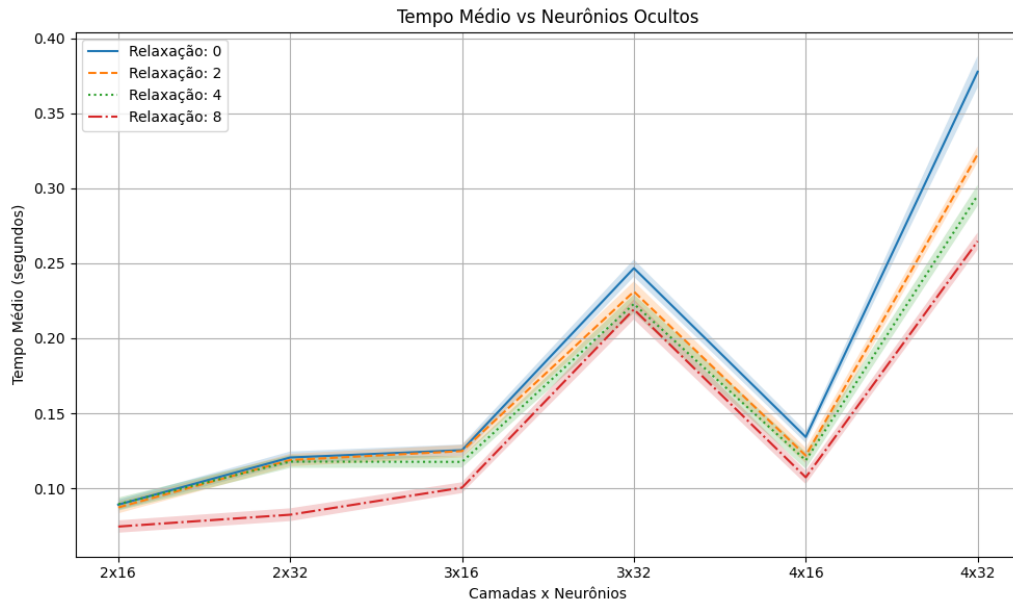


Figura 23 – Gráfico do Tempo Médio de Explicação para o conjunto de dados Iris, utilizando diferentes configurações de relaxação aplicadas às redes neurais.

Fonte: autoral

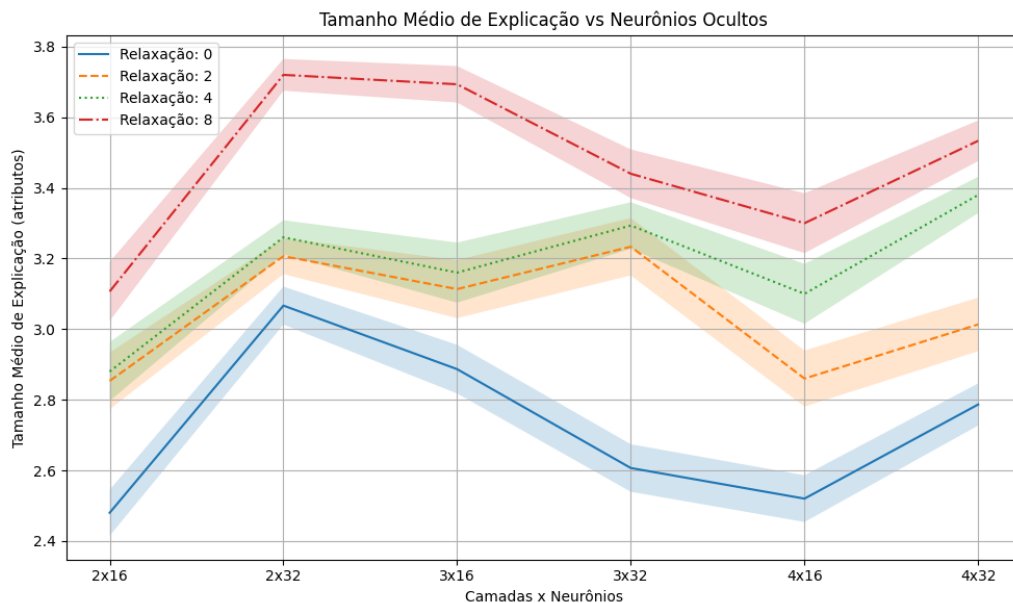


Figura 24 – Gráfico do Tamanho Médio de Explicação para o conjunto de dados Iris, utilizando diferentes configurações de relaxação aplicadas às redes neurais.

Fonte: autoral

Na Figura 24, observa-se o tamanho médio das explicações chegar ao valor máximo permitido pelo conjunto, uma vez que possui apenas 4 atributos.

5.3.5 Conjunto Wine

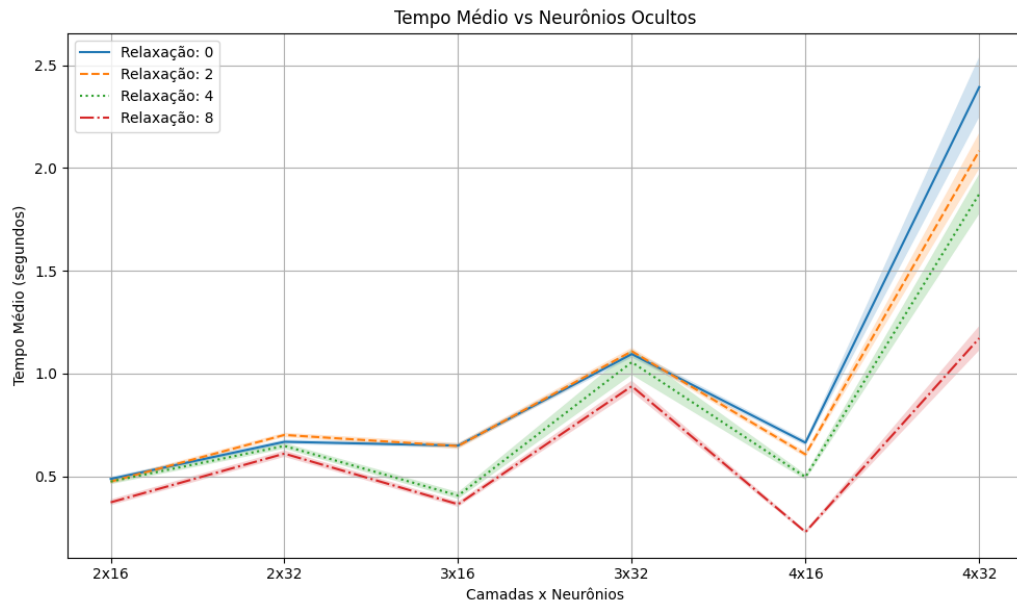


Figura 25 – Gráfico do Tempo Médio de Explicação para o conjunto de dados Wine, utilizando diferentes configurações de relaxação aplicadas às redes neurais.

Fonte: autoral

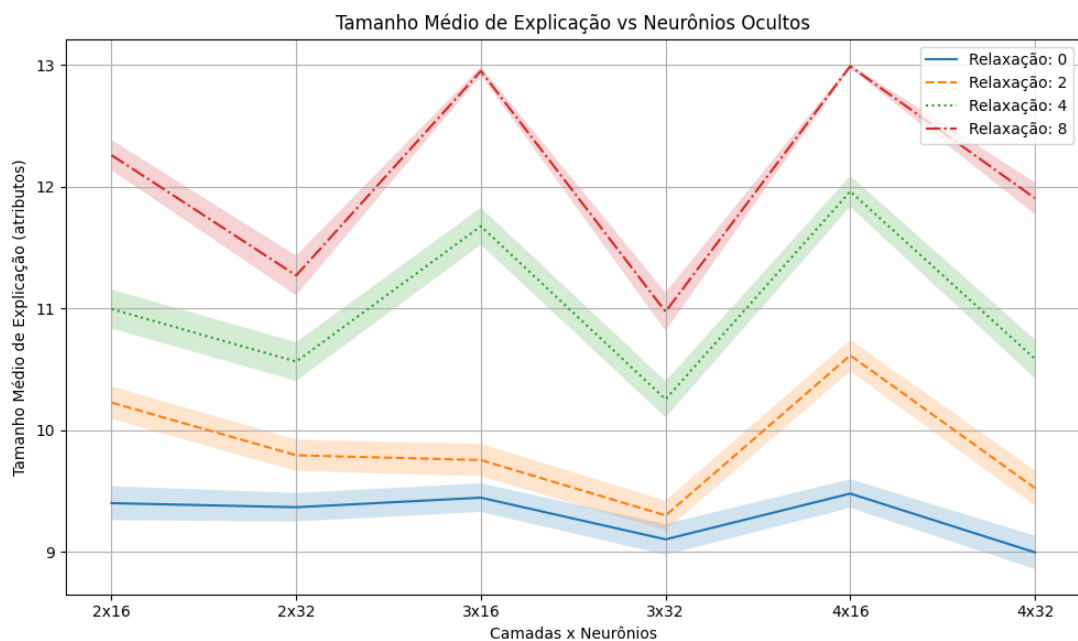


Figura 26 – Gráfico do Tamanho Médio de Explicação para o conjunto de dados Wine, utilizando diferentes configurações de relaxação aplicadas às redes neurais.

Fonte: autoral

O conjunto de dados Wine consiste em 178 instâncias e 13 atributos contínuos, referentes a propriedades químicas de vinhos provenientes de três diferentes cultivares. Sua baixa dimensionalidade o torna ideal para testes iniciais de verificação de desempenho e qualidade das explicações.

Na Figura 25 é possível observar que o relaxamento é capaz de reduzir o tempo necessário para gerar explicações em Redes com maiores quantidades de neurônios nas camadas ocultas.

A Figura 26 sugere que é necessário equilibrar o ganho de eficiência na geração de explicações com a qualidade das explicações por meio da escolha da quantidade de relaxamentos.

5.3.6 Conjunto Breast Cancer

O conjunto Breast Cancer (Wisconsin Diagnostic) contém 569 instâncias com 30 atributos contínuos, extraídos de imagens digitalizadas de células da mama. Ele é largamente utilizado em tarefas de diagnóstico médico, representando um caso realista e importante de aplicação de modelos explicáveis.

A Figura 27 revela, em contraste com os exemplos anteriores, que o impacto da relaxação é mais significativo em modelos com maior número de atributos, possivelmente devido à inclusão de um maior volume de restrições no problema.

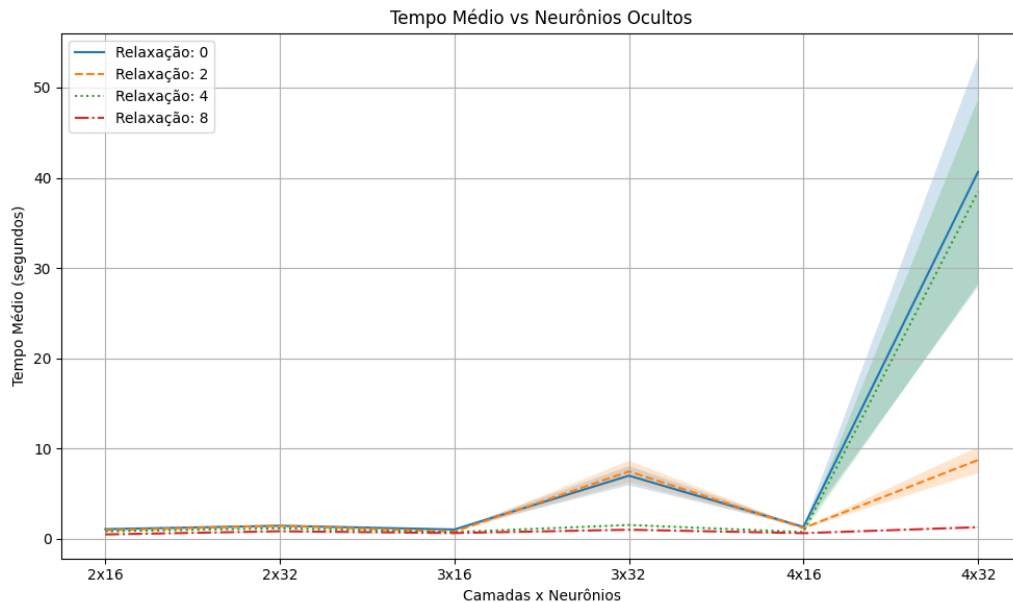


Figura 27 – Gráfico do Tempo Médio de Explicação para o conjunto de dados Breast Cancer, utilizando diferentes configurações de relaxação aplicadas às redes neurais.

Fonte: autoral

A Figura 28 mostra que, em determinados casos, o tamanho médio das explicações

pode ser mantido, o que sugere a viabilidade de uma abordagem alternativa na seleção dos neurônios a serem relaxados.

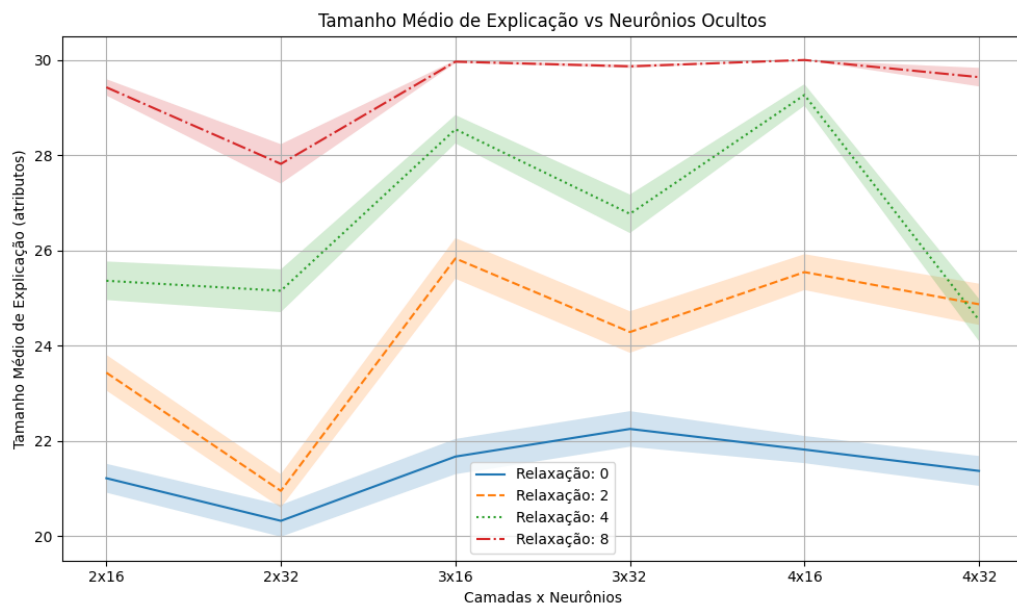


Figura 28 – Gráfico do Tamanho Médio de Explicação para o conjunto de dados Breast Cancer, utilizando diferentes configurações de relaxação aplicadas às redes neurais.

Fonte: autoral

5.3.7 Conjunto Digits

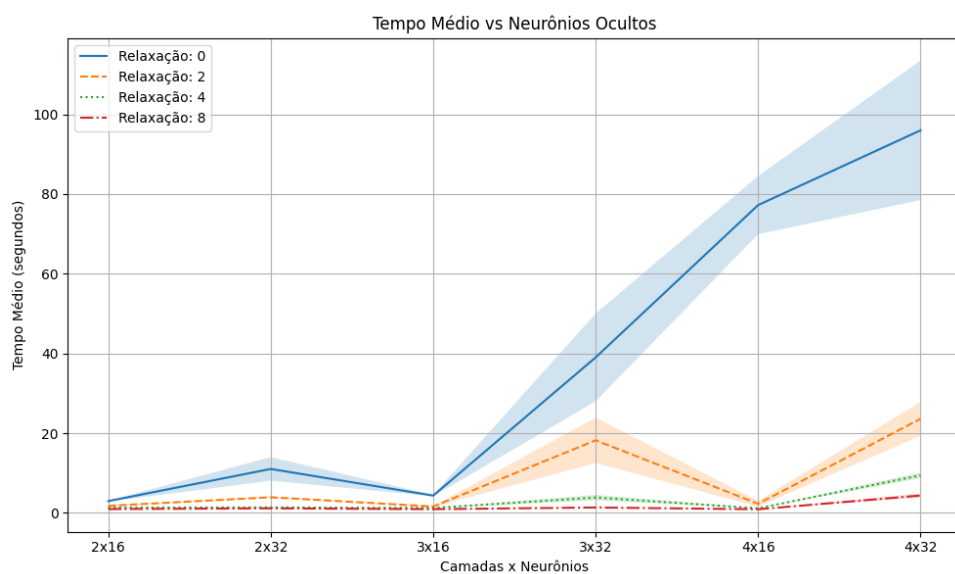


Figura 29 – Gráfico do Tempo Médio de Explicação para o conjunto de dados Digits, utilizando diferentes configurações de relaxação aplicadas às redes neurais.

Fonte: autoral

O conjunto Digits contém 1797 imagens em escala de cinza de dígitos manuscritos (de 0 a 9), com resolução de 8×8 pixels, totalizando 64 atributos por instância. Por ser uma versão reduzida do MNIST, é particularmente útil em modelos explicáveis simples de visão computacional.

A Figura 29 apresenta um cenário que corrobora o argumento de que a presença de mais atributos aumenta a complexidade em modelos maiores. Nesse conjunto, o tempo de processamento torna-se um fator crítico, evidenciando o desafio da escalabilidade.

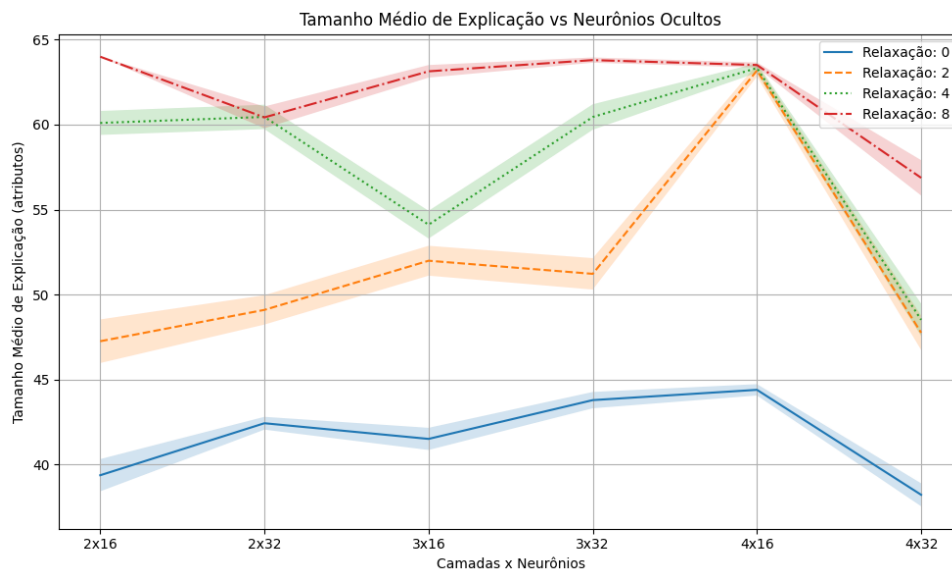


Figura 30 – Gráfico do Tamanho Médio de Explicação para o conjunto de dados Digits, utilizando diferentes configurações de relaxação aplicadas às redes neurais.

Fonte: autoral

A Figura 30 sugere que, ao selecionar criteriosamente a quantidade de neurônios a serem relaxados, é possível manter a quantidade de atributos considerados na explicação até 20% maior em comparação com modelos não relaxados.

5.3.8 Conjunto Sonar

Composto por 208 instâncias descritas por 60 atributos contínuos, o conjunto Sonar representa leituras de retorno de sinais sonares refletidos por minas ou rochas submarinas.

A Figura 31 evidencia que, devido à sua natureza ruidosa e alta dimensionalidade, o conjunto Sonar apresenta uma variação significativa no tempo de geração das explicações. Essa instabilidade pode ser atribuída à dificuldade dos modelos em identificar padrões consistentes, o que afeta a ativação dos neurônios e, consequentemente, aumenta a complexidade do processo.

A Figura 32 reforça a ideia de que a escolha adequada da quantidade de neurônios a serem relaxados permite a geração de explicações aproximadas, mantendo a garantia

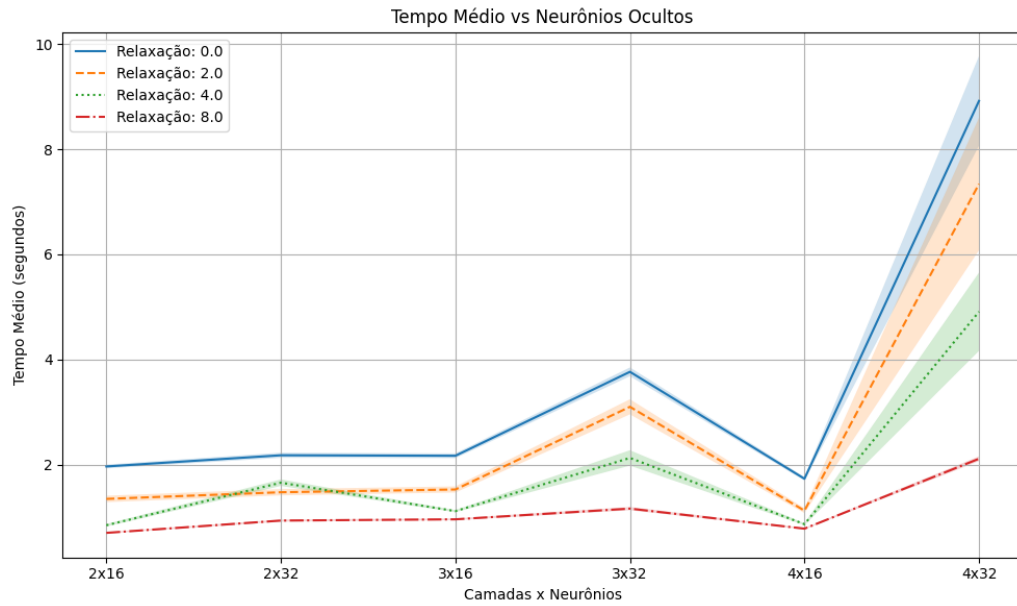


Figura 31 – Gráfico do Tempo Médio de Explicação para o conjunto de dados Sonar, utilizando diferentes configurações de relaxação aplicadas às redes neurais.

Fonte: autoral

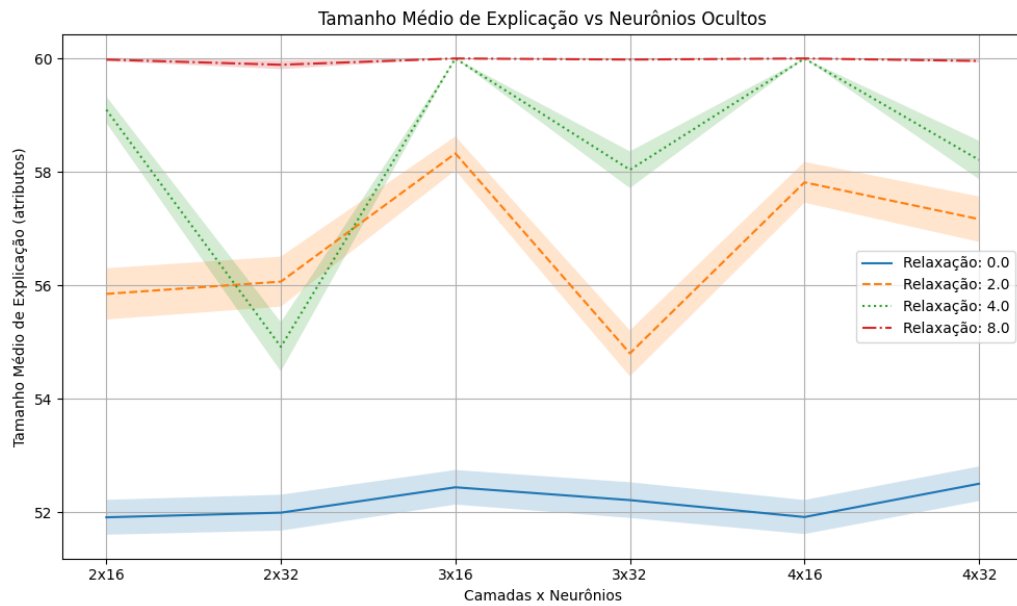


Figura 32 – Gráfico do Tamanho Médio de Explicação para o conjunto de dados Sonar, utilizando diferentes configurações de relaxação aplicadas às redes neurais.

Fonte: autoral

de corretude e, ao mesmo tempo, reduzindo o tempo de processamento em relação aos modelos não relaxados.

5.3.9 Conjunto MNIST

O MNIST (Modified National Institute of Standards and Technology) é um dos conjuntos de dados mais conhecidos no aprendizado de máquina no contexto de visão computacional, contendo 70.000 imagens de dígitos manuscritos de 28×28 pixels, totalizando 784 atributos por instância. Devido à sua natureza visual, o conjunto MNIST configura-se como um cenário e apropriado para meios complementares de verificação, validação e interpretação das explicações, o que favorece análises qualitativas e comparações visuais entre as decisões dos modelos.

A Figura 33 evidencia a eficácia do relaxamento na redução do tempo necessário para a geração de explicações, mesmo em um conjunto de imagens de alta dimensionalidade como o MNIST. Observa-se que, quanto maior o modelo, mais expressiva se torna essa redução, sugerindo que a aplicação do relaxamento pode aproximar parcialmente as explicações de uma complexidade polinomial.

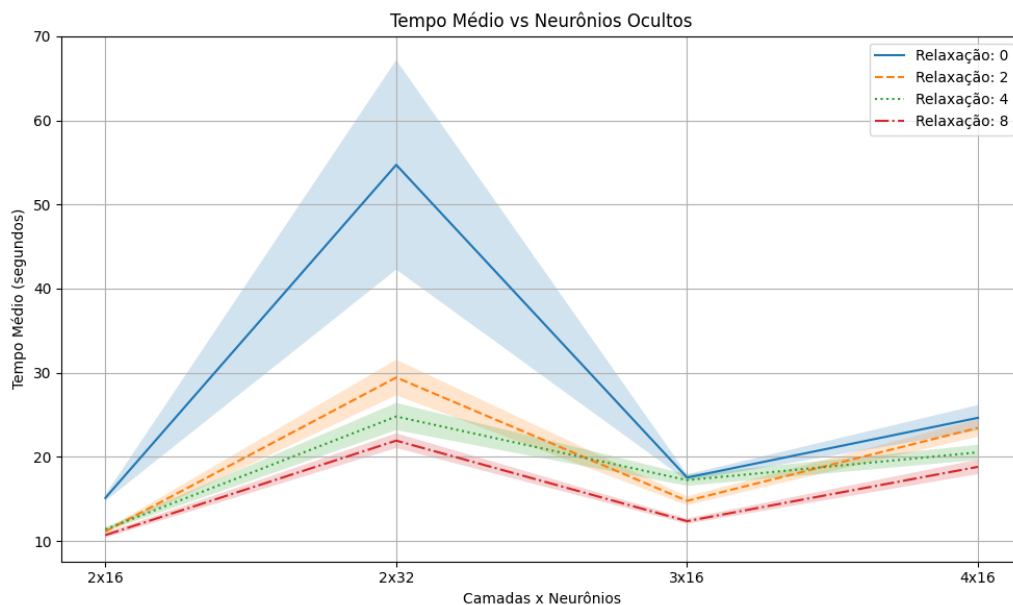


Figura 33 – Gráfico do Tempo Médio de Explicação para o conjunto de dados MNIST, utilizando diferentes configurações de relaxação aplicadas às redes neurais.

Fonte: autoral

A Figura 34 reforça que o uso do relaxamento permite gerar explicações em tempo hábil, preservando um tamanho interpretável, mesmo em conjuntos com elevada dimensionalidade.

A visualização dos *pixels* que compõem uma explicação podem ser uma ferramenta valiosa para a análise interpretável de classificadores, especialmente em tarefas envolvendo imagens, como as da base MNIST. Em contextos onde a compreensibilidade dos modelos é fundamental, como na medicina, segurança e sistemas de apoio à decisão, oferecer não

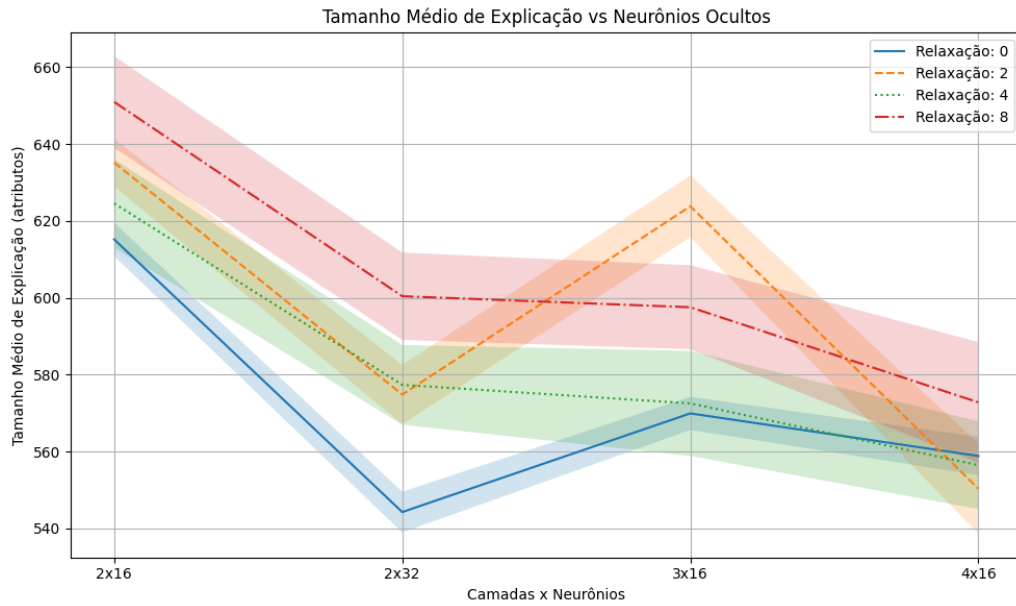


Figura 34 – Gráfico do Tamanho Médio de Explicação para o conjunto de dados MNIST, utilizando diferentes configurações de relaxação aplicadas às redes neurais.

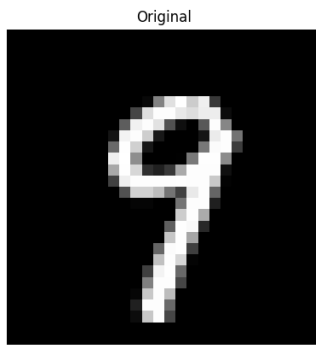
Fonte: autoral

apenas uma justificativa quantitativa, mas também visualmente intuitiva, amplia significativamente o valor e o impacto das explicações geradas.

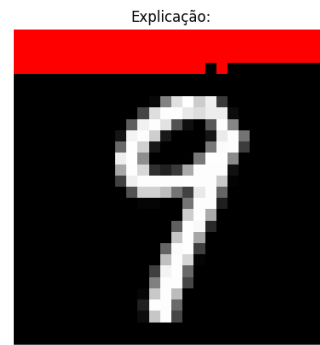
A abordagem de destacar visualmente os *pixels* mais relevantes atua como uma ponte entre o comportamento interno do modelo e a interpretação humana. Isso é particularmente útil em modelos de redes neurais profundas, que, apesar de apresentarem desempenho elevado, são frequentemente criticados por sua natureza de caixa-preta. A Figura 35 apresenta um exemplo de uma imagem com suas respectivas explicações. A Figura 35a contém a imagem original retirada do conjunto. Já as Figuras 35b, 35c e 35d contém explicações geradas a partir de um modelo de 2 camadas ocultas e 16 neurônios por camada com nenhuma relaxação, 2 relaxações por camada e 8 relaxações por camada oculta, respectivamente.

Nesse exemplo da Figura 35b é possível notar que a explicação, em sua forma mais rigorosa (sem relaxações), tende a remover os primeiros atributos avaliados, encontrando imediatamente uma explicação mínima com garantia de correteza. Essa estratégia, embora teoricamente sólida, pode levar à exclusão precoce de regiões interpretativamente relevantes da imagem, resultando em explicações menos distribuídas espacialmente.

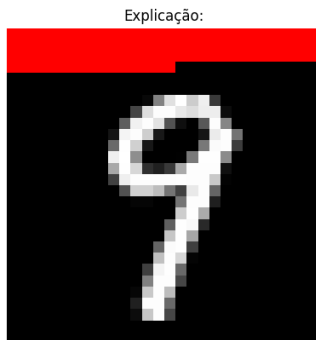
Ao introduzir um nível intermediário de relaxação como no caso de dois neurônios relaxados por camada na Figura 35c unidades relaxadas o explicador torna-se um pouco mais contido. Já não é possível remover tantos atributos de forma imediata, o que impõe uma limitação sobre o tamanho da explicação. Com isso, a explicação gerada preserva um pouco mais da estrutura original, ainda que continue centrada em regiões específicas da



(a) Exemplo de imagem original representando o número 9 em escala de cinza retirada do conjunto MNIST



(b) Exemplo da imagem do número 9 com atributos não considerados como parte de uma explicação com nenhum relaxamento destacados em vermelho.



(c) Exemplo da imagem do número 9 com atributos não considerados como parte de uma explicação com 2 relaxamentos por camada oculta destacados em vermelho.



(d) Exemplo da imagem do número 9 com atributos não considerados como parte de uma explicação com 8 relaxamentos por camada oculta destacados em vermelho.

Figura 35 – Exemplo de instância explicada retirada do conjunto MNIST.

imagem.

Porém, com oito neurônios relaxados por camada na Figura 35d, observa-se um comportamento distinto: menos atributos são considerados parte da explicação logo nas primeiras iterações. Esse “atraso” permite a verificação de atributos mais distribuídos ao longo da imagem, cobrindo diferentes regiões relevantes e, potencialmente, capturando uma representação mais rica e interpretável do raciocínio do modelo.

6 Conclusão

O estudo das redes neurais artificiais é extremamente abrangente e envolve muitas áreas do conhecimento. As aplicações práticas das RNAs permitem avanços inimagináveis há 50 anos atrás. Um exemplo disso é a medicina, que aproveita-se da capacidade das RNAs para realizar diagnósticos com muita precisão em imagens de pacientes, justificando a importância de se desenvolver métodos que produzam explicações corretas em tempo hábil.

Devido ao problema da escalabilidade observada em métodos de explicação exata aplicados a redes neurais de grande porte, torna-se essencial investigar alternativas que mitiguem esse gargalo computacional.

Nesse trabalho, nós propomos uma técnica para melhorar a eficiência de explicações de redes neurais que utiliza relaxações de variáveis binárias em modelagens de Programação Linear Inteira Mista. Essa técnica proposta parte da codificação de (CARVALHO; OLIVEIRA; ROCHA, 2024), com a aplicação de relaxação em algumas variáveis binárias para fornecer explicações conforme o algoritmo de Ignatiev, Narodytska e Marques-Silva (2019).

A técnica de relaxação, nesse contexto, surge como uma abordagem promissora ao permitir uma redução significativa no tempo de geração das explicações, mantendo, em muitos casos, um nível aceitável de concisão das explicações. Embora essa estratégia possa introduzir certo afastamento da minimalidade ideal, os ganhos em eficiência demonstrados experimentalmente justificam seu uso, especialmente em cenários em que o tempo de resposta é um fator crítico.

Além disso, a técnica empregada demonstrou, nos experimentos, proporcionar uma redução de até 59.9% no tempo médio de explicação para um aumento de apenas 10.32% no tamanho médio da explicação. Esse resultado evidencia o potencial da relaxação como estratégia eficiente para acelerar o processo de geração de explicações em modelos maiores.

Para trabalhos futuros, recomenda-se uma investigação mais aprofundada sobre critérios de seleção das variáveis a serem relaxadas. Uma escolha mais criteriosa pode contribuir para melhorar ainda mais a eficiência computacional, sem comprometer significativamente a minimalidade da explicação. Avaliar diferentes estratégias de relaxação, possivelmente guiadas por medidas de ativação dos neurônios, pode abrir caminhos para um equilíbrio mais vantajoso entre custo computacional e minimalidade explicativa. Assim, o aprimoramento de mecanismos que escolham de forma inteligente quais variáveis podem ser relaxadas representa um caminho relevante para ampliar a aplicabilidade prática das explicações formais em contextos mais complexos.

Referências

- ABIODUN, O. I. et al. State-of-the-art in artificial neural network applications: A survey. *Heliyon*, Elsevier, v. 4, n. 11, p. e00938, 2018. Citado na página 27.
- ALEMANY, J.; KASPRZYK, L.; MAGNAGO, F. Effects of binary variables in mixed integer linear programming based unit commitment in large-scale electricity markets. *Electric Power Systems Research*, Elsevier, v. 160, p. 429–438, 2018. Citado na página 38.
- ALGHOUL, A. et al. Email classification using artificial neural network. *IJARW*, 2018. Citado 2 vezes nas páginas 27 e 45.
- ANDERSON, B. G. et al. Tightened convex relaxations for neural network robustness certification. In: IEEE. *2020 59th IEEE Conference on Decision and Control (CDC)*. [S.l.], 2020. p. 2190–2197. Citado na página 38.
- ASHA, R.; KR, S. K. Credit card fraud detection using artificial neural network. *Global Transitions Proceedings*, Elsevier, v. 2, n. 1, p. 35–41, 2021. Citado na página 27.
- BELOTTI, P. et al. On handling indicator constraints in mixed integer programming. *Computational Optimization and Applications*, Springer, v. 65, n. 3, p. 545–566, 2016. Citado na página 28.
- BÉNICHOU, M. et al. Experiments in mixed-integer linear programming. *Mathematical Programming*, Springer, v. 1, n. 1, p. 76–94, 1971. Citado na página 36.
- BEST, M. J.; RITTER, K. *Linear programming*. [S.l.]: Prentice Hall Englewood Cliffs, NY, 1985. Citado na página 35.
- BIERE, A.; HEULE, M.; MAAREN, H. van. *Handbook of satisfiability*. [S.l.]: IOS press, 2009. v. 185. Citado na página 39.
- CADAMURO, G.; GILAD-BACHRACH, R.; ZHU, X. Debugging machine learning models. v. 103, 2016. Citado na página 27.
- CARVALHO, L. C.; OLIVEIRA, S. A.; ROCHA, T. A. Comparing neural network encodings for logic-based explainability. In: SPRINGER. *Brazilian Conference on Intelligent Systems*. [S.l.], 2024. p. 281–295. Citado 4 vezes nas páginas 28, 45, 53 e 79.
- ELBOHER, Y. Y.; GOTTSCHLICH, J.; KATZ, G. An abstraction-based framework for neural network verification. p. 43–65, 2020. Citado na página 27.
- ELDAN, R.; SHAMIR, O. The power of depth for feedforward neural networks. In: FELDMAN, V.; RAKHLIN, A.; SHAMIR, O. (Ed.). *29th Annual Conference on Learning Theory*. Columbia University, New York, New York, USA: PMLR, 2016. (Proceedings of Machine Learning Research, v. 49), p. 907–940. Disponível em: <<https://proceedings.mlr.press/v49/eldan16.html>>. Citado na página 31.
- FAÍSCA, N. P. et al. Global optimization of multi-parametric milp problems. *Journal of Global Optimization*, Springer, v. 45, p. 131–151, 2009. Citado na página 46.

FERNANDO, L. *Relaxing ReLU on ANNs for Explications*. 2022. <<https://github.com/Luizfernandopq/XAI-nn-relu.git>>. Citado na página 61.

FISCHETTI, M.; JO, J. Deep neural networks and mixed integer linear optimization. *Constraints*, Springer, v. 23, n. 3, p. 296–309, 2018. Citado na página 27.

GRIMSTAD, B.; ANDERSSON, H. Relu networks as surrogate models in mixed-integer linear programs. *Computers & Chemical Engineering*, Elsevier, v. 131, p. 106580, 2019. Citado na página 52.

HILL, R. K. What an algorithm is. *Philosophy & Technology*, Springer, v. 29, n. 1, p. 35–59, 2016. Citado na página 27.

HUANG, G.-B.; CHEN, Y.-Q.; BABRI, H. Classification ability of single hidden layer feedforward neural networks. *IEEE Transactions on Neural Networks*, v. 11, n. 3, p. 799–801, 2000. Citado na página 31.

IGNATIEV, A. Towards trustable explainable ai. In: BESSIERE, C. (Ed.). *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*. International Joint Conferences on Artificial Intelligence Organization, 2020. p. 5154–5158. Early Career. Disponível em: <<https://doi.org/10.24963/ijcai.2020/726>>. Citado na página 28.

IGNATIEV, A.; NARODYTSKA, N.; MARQUES-SILVA, J. Abduction-based explanations for machine learning models. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. [S.l.: s.n.], 2019. v. 33, n. 01, p. 1511–1519. Citado 6 vezes nas páginas 27, 28, 40, 45, 49 e 79.

KARMARKAR, N. A new polynomial-time algorithm for linear programming. In: *Proceedings of the sixteenth annual ACM symposium on Theory of computing*. [S.l.: s.n.], 1984. p. 302–311. Citado na página 28.

KATZ, G. et al. Reluplex: An efficient smt solver for verifying deep neural networks. In: SPRINGER. *International conference on computer aided verification*. [S.l.], 2017. p. 97–117. Citado na página 28.

KOCH, T. et al. Progress in mathematical programming solvers from 2001 to 2020. *EURO Journal on Computational Optimization*, Elsevier, v. 10, p. 100031, 2022. Citado na página 28.

KOH, P. W.; LIANG, P. Understanding black-box predictions via influence functions. In: PMLR. *International conference on machine learning*. [S.l.], 2017. p. 1885–1894. Citado na página 27.

KROENING, D.; STRICHMAN, O. *Decision procedures*. [S.l.]: Springer, 2016. Citado na página 39.

LECUN, Y.; CORTES, C.; BURGESS, C. J. *The MNIST Database of Handwritten Digits*. 1998. <<http://yann.lecun.com/exdb/mnist/>>. Citado na página 60.

LEUNG, H.; HAYKIN, S. The complex backpropagation algorithm. *IEEE Transactions on Signal Processing*, v. 39, n. 9, p. 2101–2104, 1991. Citado na página 31.

- LIASHCHYNSKYI, P.; LIASHCHYNSKYI, P. Grid search, random search, genetic algorithm: a big comparison for nas. *arXiv preprint arXiv:1912.06059*, 2019. Citado na página 61.
- LIBERTI, L. Continuous reformulation of binary variables, revisited. In: SPRINGER. *International Conference on Mathematical Optimization Theory and Operations Research*. [S.l.], 2021. p. 201–215. Citado na página 38.
- LIU, C. et al. Algorithms for verifying deep neural networks. *Foundations and Trends® in Optimization*, Now Publishers, Inc., v. 4, n. 3-4, p. 244–404, 2021. Citado na página 27.
- LIU, S. et al. Investigation of a transformer-based hybrid artificial neural networks for climate data prediction and analysis. *Frontiers in Environmental Science*, Frontiers Media SA, v. 12, p. 1464241, 2025. Citado na página 27.
- LYASHENKO, V. et al. Recognition of voice commands based on neural network. *TEM Journal*, 2021. Citado na página 27.
- Markelle Kelly, Rachel Longjohn, Kolby Nottingham. *The UCI Machine Learning Repository*. 2025. <<https://archive.ics.uci.edu/datasets/>>. Acesso em: 25 jul. 2025. Citado na página 59.
- MARQUES-SILVA, J. Logic-based explainability in machine learning. In: *Reasoning Web. Causality, Explanations and Declarative Knowledge: 18th International Summer School 2022, Berlin, Germany, September 27–30, 2022, Tutorial Lectures*. [S.l.]: Springer, 2023. p. 24–104. Citado na página 28.
- MERCIONI, M. A.; HOLBAN, S. The most used activation functions: Classic versus current. In: IEEE. *2020 International Conference on Development and Application Systems (DAS)*. [S.l.], 2020. p. 141–145. Citado na página 34.
- METZ, J. Interpretação de clusters gerados por algoritmos de clustering hierárquico. *Teste*, 2006. Citado na página 27.
- MUSLEH, M. M. et al. Predicting liver patients using artificial neural network. *International Journal of Academic Information Systems Research (IJAIRS)*, v. 3, n. 10, 2019. Citado 2 vezes nas páginas 27 e 45.
- PASZKE, A. et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, v. 32, 2019. Citado na página 59.
- PEDREGOSA, F. et al. *Scikit-learn: Machine Learning in Python*. 2011. 2825–2830 p. Disponível em: <<https://scikit-learn.org/>>. Citado na página 60.
- PIA, A. D.; KHAJAVIRAD, A. The complete edge relaxation for binary polynomial optimization. *arXiv preprint arXiv:2507.12831*, 2025. Citado na página 37.
- RAUBER, T. W. Redes neurais artificiais. *Universidade Federal do Espírito Santo*, p. 29, 2005. Citado na página 27.
- RIBEIRO, M. T.; SINGH, S.; GUESTIN, C. "why should i trust you?" explaining the predictions of any classifier. In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. [S.l.: s.n.], 2016. p. 1135–1144. Citado na página 27.

- ROCHA, M.; CORTEZ, P.; NEVES, J. Evolution of neural networks for classification and regression. *Neurocomputing*, v. 70, n. 16, p. 2809–2816, 2007. ISSN 0925-2312. Neural Network Applications in Electrical Engineering Selected papers from the 3rd International Work-Conference on Artificial Neural Networks (IWANN 2005). Disponível em: <<https://www.sciencedirect.com/science/article/pii/S0925231207001567>>. Citado na página 32.
- SHAPIRO, S.; KISSEL, T. K. Classical logic. *Stanford encyclopedia of philosophy*, 2018. Citado na página 40.
- SMITH, J. C.; TASKIN, Z. C. A tutorial guide to mixed-integer programming models and solution techniques. *Optimization in medicine and biology*, London: Auerbach Publications, p. 521–548, 2008. Citado na página 37.
- TJENG, V.; XIAO, K.; TEDRAKE, R. Evaluating robustness of neural networks with mixed integer programming. *arXiv preprint arXiv:1711.07356*, 2017. Citado 4 vezes nas páginas 27, 41, 44 e 45.
- VENZKE, A.; CHATZIVASILEIADIS, S. Verification of neural network behaviour: Formal guarantees for power system applications. *IEEE Transactions on Smart Grid*, IEEE, v. 12, n. 1, p. 383–397, 2020. Citado na página 27.
- WENG, T.-W. et al. Evaluating the robustness of neural networks: An extreme value theory approach. *arXiv preprint arXiv:1801.10578*, 2018. Citado na página 27.
- YOU, J. et al. Graph structure of neural networks. In: III, H. D.; SINGH, A. (Ed.). *Proceedings of the 37th International Conference on Machine Learning*. PMLR, 2020. (Proceedings of Machine Learning Research, v. 119), p. 10881–10891. Disponível em: <<https://proceedings.mlr.press/v119/you20b.html>>. Citado na página 31.