

Victor Patrick Sena Barbosa Lima

# Comparação entre Modelos de Redes Neurais U-Net e U-Net Fuzzy para Mensurar Regiões de Imagens de Ultrassonografia Bovina



UNIVERSIDADE FEDERAL DE UBERLÂNDIA  
FACULDADE DE MATEMÁTICA

2026

Victor Patrick Sena Barbosa Lima

# Comparação entre Modelos de Redes Neurais U-Net e U-Net Fuzzy para Mensurar Regiões de Imagens de Ultrassonografia Bovina

**Dissertação** apresentada ao Programa de Pós-Graduação em Matemática da Universidade Federal de Uberlândia, como parte dos requisitos para obtenção do título de **Mestre em Matemática**.

**Área de Concentração:** Matemática.

**Linha de Pesquisa:** Matemática Aplicada.

**Orientadora:** Profa. Dra. Rosana Sueli da Motta Jafelice.

**Coorientador:** Prof. Dr. Caio Augusto Rodrigues dos Santos.

UBERLÂNDIA - MG

2026

Ficha Catalográfica Online do Sistema de Bibliotecas da UFU  
com dados informados pelo(a) próprio(a) autor(a).

L732  
2026

Lima, Victor Patrick Sena Barbosa, 1999-  
Comparação entre Modelos de Redes Neurais U-Net e U-Net  
Fuzzy para Mensurar Regiões de Imagens de Ultrassonografia  
Bovina [recurso eletrônico] / Victor Patrick Sena Barbosa Lima. -  
2026.

Orientadora: Rosana Sueli da Motta Jafelice.

Coorientadora: Caio Augusto Rodrigues dos Santos.

Dissertação (Mestrado) - Universidade Federal de Uberlândia,  
Pós-graduação em Matemática.

Modo de acesso: Internet.

DOI <http://doi.org/10.14393/ufu.di.2026.523>

Inclui bibliografia.

Inclui ilustrações.

1. Matemática. I. Jafelice, Rosana Sueli da Motta, 1964-,  
(Orient.). II. Santos, Caio Augusto Rodrigues dos, 1989-, (Coorient.).  
III. Universidade Federal de Uberlândia. Pós-graduação em  
Matemática. IV. Título.

CDU: 51

Bibliotecários responsáveis pela estrutura de acordo com o AACR2:

Gizele Cristine Nunes do Couto - CRB6/2091

Nelson Marcos Ferreira - CRB6/3074



## **ATA DE DEFESA - PÓS-GRADUAÇÃO**

|                                    |                                                                                                                            |                 |       |                       |       |
|------------------------------------|----------------------------------------------------------------------------------------------------------------------------|-----------------|-------|-----------------------|-------|
| Programa de Pós-Graduação em:      | Matemática                                                                                                                 |                 |       |                       |       |
| Defesa de:                         | Dissertação de Mestrado Acadêmico, 137, PPGMAT.                                                                            |                 |       |                       |       |
| Data:                              | 28/07/2026                                                                                                                 | Hora de início: | 15:00 | Hora de encerramento: | 17:00 |
| Matrícula do Discente:             | 12422MAT006                                                                                                                |                 |       |                       |       |
| Nome do Discente:                  | Victor Patrick Sena Barbosa Lima                                                                                           |                 |       |                       |       |
| Título do Trabalho:                | Comparação entre Modelos de Redes Neurais U-Net e U-Net Fuzzy para Mensurar Regiões de Imagens de Ultrassonografia Bovina. |                 |       |                       |       |
| Área de concentração:              | Matemática                                                                                                                 |                 |       |                       |       |
| Linha de pesquisa:                 | Matemática Aplicada                                                                                                        |                 |       |                       |       |
| Projeto de Pesquisa de vinculação: | Estudo Comparativo de Modelos Matemáticos Preditivos através dos Conjuntos Fuzzy                                           |                 |       |                       |       |

Reuniu-se na Sala 1F 119 (Sala Multiuso do Instituto de Matemática e Estatística) - Bloco 1F (Campus Santa Mônica) da Universidade Federal de Uberlândia e por videoconferência, a Banca Examinadora, designada pelo Colegiado do Programa de Pós-graduação em Matemática, assim composta: Professores Doutores: Marilaine Colnago - Universidade Estadual Paulista/UNESP, José Waldemar da Silva - IME/UFU e Rosana Sueli da Motta Jafelice - IME/UFU orientadora do candidato.

Iniciando os trabalhos a presidente da mesa, Dra. Rosana Sueli da Motta Jafelice, apresentou a Comissão Examinadora e o candidato, agradeceu a presença do público, e concedeu ao discente a palavra para a exposição do seu trabalho.

A duração da apresentação do discente e o tempo de arguição e resposta foram conforme as normas do Programa. A seguir a senhora presidente concedeu a palavra, pela ordem sucessivamente, aos examinadores, que passaram a arguir o candidato. Ultimada a arguição, que se desenvolveu dentro dos termos regimentais, a Banca, em sessão secreta, atribuiu o resultado final, considerando o candidato:

Aprovado.

Esta defesa faz parte dos requisitos necessários à obtenção do título de Mestre.

O competente diploma será expedido após cumprimento dos demais requisitos, conforme as normas do Programa, a legislação pertinente e a regulamentação interna da UFU.

Nada mais havendo a tratar foram encerrados os trabalhos. Foi lavrada a presente ata que após lida e achada conforme foi assinada pela Banca Examinadora.



Documento assinado eletronicamente por **Marilaine Colnago, Usuário Externo**, em 28/07/2026, às 17:14, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do [Decreto nº 8.539, de 8 de outubro de 2015](#).



Documento assinado eletronicamente por **Rosana Sueli da Motta Jafelice, Professor(a) do Magistério Superior**, em 28/07/2026, às 17:18, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do [Decreto nº 8.539, de 8 de outubro de 2015](#).



Documento assinado eletronicamente por **José Waldemar da Silva, Professor(a) do Magistério Superior**, em 28/07/2026, às 17:23, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do [Decreto nº 8.539, de 8 de outubro de 2015](#).



A autenticidade deste documento pode ser conferida no site [https://www.sei.ufu.br/sei/controlador\\_externo.php?acao=documento\\_conferir&id\\_orgao\\_acesso\\_externo=0](https://www.sei.ufu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0), informando o código verificador **7477159** e o código CRC **938B2420**.

## Dedicatória

*Dedico este trabalho à minha família, que sempre me apoiou e incentivou em minha jornada acadêmica. Sem o amor e o suporte de vocês, esta conquista não teria sido possível.*

## Agradecimentos

Gostaria de agradecer nominalmente a todos que, de alguma forma, contribuíram para a realização deste trabalho. Parafraseando Sir. Isaac Newton, só estou aqui porque me apoiei nos ombros de gigantes. Na impossibilidade de agradecer a todos, por serem muitos, quero mencionar, ao menos, quatro grupos a quem fiquei especialmente grato.

À minha família, meus pais, pelo apoio, incentivo e motivação ao longo de toda essa caminhada e minhas irmãs, que sempre estiveram oferecendo suporte e carinho. Agradeço a Jéssica Plífinar e Cecília Plífinar, meus amores, que estiveram comigo durante essa jornada, pelo companheirismo e pelos ensinamentos.

Aos meus professores, especialmente à minha orientadora, professora Dra. Rosana S. M. Jafelice, e ao meu coorientador, professor Dr. Caio A. R. Santos, pela orientação, pelo apoio e por me ajudarem a superar os desafios encontrados durante o desenvolvimento deste trabalho.

Agradeço aos colaboradores, que disponibilizaram todo o banco de dados, professora Dra. Amanda M. Maiorano e as pesquisadoras do Centro Avançado de Pesquisa e Desenvolvimento de Bovinos de Corte, Maria E. Z. Mercadante e Karine D. S. Pinto.

Por fim, agradeço à CAPES pelo apoio financeiro. A todos que de alguma forma contribuíram para a realização deste trabalho, meu muito obrigado.

## Resumo

A pecuária bovina é um dos pilares fundamentais da economia brasileira. A ultrassonografia de carcaça permite mensurar, *in vivo*, a Área de Olho de Lombo (AOL) e as Espessuras de Gordura Subcutânea do Lombo (EGL) e da Garupa (EGG), características relacionadas à musculosidade e desenvolvimento precoce de bovinos. Entretanto, a obtenção dessas medidas ainda depende, predominantemente, da delimitação manual e interpretação realizadas por especialistas, um processo subjetivo, demorado e suscetível à variabilidade entre avaliadores. O objetivo deste trabalho é desenvolver e comparar dois modelos para delimitar e mensurar automaticamente esses indicadores em imagens de ultrassom de duas raças bovinas. O primeiro modelo é uma rede neural convolucional U-Net com abordagem Multitarefa e o segundo, U-Net Fuzzy, é uma U-Net integrada ao Sistema de Inferência Neuro-Fuzzy Adaptativo, em inglês *Adaptive Neuro-Fuzzy Inference System* (ANFIS). As imagens foram obtidas de animais vivos do Centro Avançado de Pesquisa e Desenvolvimento de Bovinos de Corte, Instituto de Zootecnia, SAA-SA, localizado em Sertãozinho - SP, totalizando 135 imagens, das quais 108 são da raça Nelore e 27 da raça Caracu, que foram ampliadas por técnicas de expansão de dados, como a inclusão das mesmas imagens com redução e aumento de brilho. As máscaras de segmentação utilizadas como referência, conhecidas como *ground truth*, foram criadas manualmente com o auxílio de uma especialista, consistindo em imagens binárias utilizadas para isolar regiões específicas dentro da imagem, servindo como referência para o treinamento e avaliação dos modelos. O desempenho dos modelos são avaliados por meio de métricas como acurácia, erro absoluto médio, em inglês *Mean Absolute Error* (MAE) e o coeficiente de Dice, que quantifica a similaridade entre a segmentação produzida pelo modelo e as máscaras de segmentação desenvolvidas pela especialista. Os resultados indicam que o modelo U-Net Fuzzy é superior ao modelo U-Net Multitarefa para mensurar os indicadores AOL e EGG em todas as métricas de avaliação, com destaque para a acurácia de 95,15% na AOL e 99,95% na EGL. Em contrapartida, tanto a U-Net Multitarefa quanto a U-Net Fuzzy apresentam desempenho semelhante para o indicador EGG. Por fim, foi desenvolvido um aplicativo computacional capaz de detectar regiões específicas e medir automaticamente os indicadores de interesse em imagens de ultrassom de bovinos utilizando ambos os modelos, podendo auxiliar nos processos de seleção e manejo dos rebanhos.

**Palavras-chave:** Ultrassonografia bovina; Delimitação de imagens; Redes neurais; U-Net; ANFIS.

# Abstract

Beef cattle production is one of the fundamental pillars of the Brazilian economy. Carcass ultrasonography enables the *in vivo* measurement of the Longissimus Muscle Area (LMA) and the Backfat Thickness (BFT) and Rump Fat Thickness (RFT), traits associated with muscularity and early development in cattle. However, obtaining these measurements still relies predominantly on manual delineation and interpretation performed by specialists, making the process subjective, time-consuming, and susceptible to inter-observer variability. The objective of this study is to develop and compare two models for the automatic segmentation and measurement of these indicators in ultrasound images from two cattle breeds. The first model is a multitask U-Net convolutional neural network, while the second, termed Fuzzy U-Net, integrates a U-Net architecture with an Adaptive Neuro-Fuzzy Inference System (ANFIS). The images were obtained from live animals at the Advanced Center for Beef Cattle Research and Development of the Institute of Animal Science (SAA-SP), located in Sertãozinho, São Paulo, Brazil. A total of 135 images were collected, including 108 from the Nelore breed and 27 from the Caracu breed. The dataset was expanded through data augmentation techniques, such as the inclusion of images with increased and decreased brightness. The segmentation masks used as reference, known as ground truth masks, were manually created with the assistance of a specialist. These masks consisted of binary images used to isolate specific regions within the ultrasound images, serving as references for model training and evaluation. Model performance was assessed using metrics such as accuracy, Mean Absolute Error (MAE), and the Dice coefficient, which quantifies the similarity between the segmentation generated by the model and the reference masks produced by the specialist. The results indicate that the Fuzzy U-Net model outperforms the Multitask U-Net model in measuring the LMA and BFT indicators across all evaluation metrics, achieving accuracies of 95.15% for LMA and 99.95% for BFT. In contrast, both the Multitask U-Net and Fuzzy U-Net models exhibited similar performance for the RFT indicator. Finally, a computational application was developed to automatically detect specific regions and measure the indicators of interest in bovine ultrasound images using both models. This tool has the potential to assist in cattle selection and herd management.

**Keywords:** Bovine ultrasonography; Image delineation; Neural networks; U-Net; ANFIS.

# Lista de Figuras

|      |                                                                                                                                                                         |    |
|------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 1    | Ilustração da localização das regiões de interesse. . . . .                                                                                                             | 18 |
| 2    | Imagens de ultrassom das regiões de interesse. . . . .                                                                                                                  | 19 |
| 3    | Exemplo de <i>ground truth</i> . . . . .                                                                                                                                | 20 |
| 4    | Fluxo de treinamento dos modelos. . . . .                                                                                                                               | 23 |
| 1.1  | Estrutura de um neurônio típico. . . . .                                                                                                                                | 27 |
| 1.2  | Modelo de uma unidade de McCulloch e Pitts. . . . .                                                                                                                     | 28 |
| 1.3  | Gráficos das funções de ativação. . . . .                                                                                                                               | 30 |
| 1.4  | Representação da taxa de aprendizagem. . . . .                                                                                                                          | 33 |
| 1.5  | Estrutura visual básica de uma unidade e uma rede neural com duas camadas ocultas. . . . .                                                                              | 37 |
| 1.6  | Ilustração do processo de <i>backpropagation</i> em uma rede neural. . . . .                                                                                            | 38 |
| 1.7  | Comparação entre os métodos de GD. . . . .                                                                                                                              | 40 |
| 1.8  | Ilustração do <i>overfitting</i> . . . . .                                                                                                                              | 41 |
| 1.9  | Ilustração do <i>Dropout</i> removendo 45% das unidades aleatoriamente durante uma época de treinamento. . . . .                                                        | 43 |
| 1.10 | Ilustração do <i>Early Stopping</i> . . . . .                                                                                                                           | 43 |
| 1.11 | Gráfico da derivada da função sigmoid logística. . . . .                                                                                                                | 45 |
| 1.12 | Arquiteturas de redes neurais. . . . .                                                                                                                                  | 47 |
| 1.13 | Fluxo de dados durante o processo de treinamento e avaliação da rede neural. . . . .                                                                                    | 48 |
| 1.14 | Exemplos de dígitos manuscritos do conjunto de dados MNIST. . . . .                                                                                                     | 48 |
| 1.15 | Arquitetura da rede neural para classificação de dígitos manuscritos. . . . .                                                                                           | 49 |
| 1.16 | Paraboloide hiperbólico. . . . .                                                                                                                                        | 50 |
| 1.17 | Gráfico da superfície e aproximação da superfície obtida pela MLP e a projeção da superfície do erro absoluto no plano $xy$ . . . . .                                   | 51 |
| 1.18 | Exemplo de uma imagem/matriz $1 \times 10 \times 10 \times 1$ em escala de cinza, em que cada pixel tem um valor de intensidade entre 0 (preto) e 255 (branco). . . . . | 53 |
| 1.19 | Ilustração da operação de convolução discreta. . . . .                                                                                                                  | 56 |
| 1.20 | Exemplo de convolução bidimensional. . . . .                                                                                                                            | 58 |
| 1.21 | Aplicações de diferentes <i>kernels</i> em uma imagem. . . . .                                                                                                          | 59 |

|      |                                                                                                                                                                                                                                                                                          |     |
|------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| 1.22 | Aplicação de <i>padding</i> em uma imagem. Em (a) a imagem original passou por zero <i>padding</i> . Em (b) a imagem original passou por <i>reflection padding</i> . A imagem em (c) passou por um <i>padding</i> constante, semelhante em (a), mas com valor diferente de zero. . . . . | 61  |
| 1.23 | Exemplo de <i>Max pooling</i> com dimensões $2 \times 2$ . . . . .                                                                                                                                                                                                                       | 62  |
| 1.24 | Arquitetura típica de uma Rede Neural Convolutacional com uma camada de convolução e uma camada de <i>pooling</i> . . . . .                                                                                                                                                              | 64  |
| 1.25 | Exemplo da operação de convolução transposta. . . . .                                                                                                                                                                                                                                    | 65  |
| 2.1  | Função de pertinência $\mu_A(x)$ para o conjunto fuzzy “temperatura alta”. . . . .                                                                                                                                                                                                       | 67  |
| 2.2  | Operações entre conjuntos fuzzy. . . . .                                                                                                                                                                                                                                                 | 69  |
| 2.3  | Função de pertinência de um número fuzzy trapezoidal. . . . .                                                                                                                                                                                                                            | 72  |
| 2.4  | Função de pertinência de um número fuzzy gaussiano. . . . .                                                                                                                                                                                                                              | 72  |
| 2.5  | Arquitetura de um Sistema Baseado em Regras Fuzzy (SBRF). . . . .                                                                                                                                                                                                                        | 74  |
| 2.6  | Exemplo de inferência fuzzy utilizando o método de Takagi-Sugeno. . . . .                                                                                                                                                                                                                | 75  |
| 2.7  | Arquitetura do ANFIS . . . . .                                                                                                                                                                                                                                                           | 80  |
| 2.8  | Fluxo do treinamento do ANFIS. . . . .                                                                                                                                                                                                                                                   | 81  |
| 3.1  | Imagem de entrada $X$ e <i>ground truth</i> . . . . .                                                                                                                                                                                                                                    | 83  |
| 3.2  | Aplicação de um <i>kernel</i> de convolução na imagem de entrada $X$ . . . . .                                                                                                                                                                                                           | 84  |
| 3.3  | Aplicação de <i>padding</i> na imagem de entrada $X$ . . . . .                                                                                                                                                                                                                           | 84  |
| 3.4  | Aplicação da função de ativação ReLU no mapa de características. . . . .                                                                                                                                                                                                                 | 85  |
| 3.5  | Aplicação de uma camada de <i>Max pooling</i> no mapa de características resultante após a ReLU. . . . .                                                                                                                                                                                 | 85  |
| 3.6  | Aplicação da convolução transposta no mapa de <i>pooling</i> . . . . .                                                                                                                                                                                                                   | 86  |
| 3.7  | Estrutura simplificada em forma de U da arquitetura U-Net. . . . .                                                                                                                                                                                                                       | 86  |
| 3.8  | Exemplo de conexões de atalho ( <i>skip connections</i> ) na arquitetura U-Net. . . . .                                                                                                                                                                                                  | 87  |
| 3.9  | Exemplo da compatibilização de dimensões para a concatenação. . . . .                                                                                                                                                                                                                    | 87  |
| 3.10 | Exemplo de concatenação de dois mapas de características ao longo do eixo dos canais. . . . .                                                                                                                                                                                            | 88  |
| 3.11 | Aplicação da última camada de convolução para gerar a saída da rede. . . . .                                                                                                                                                                                                             | 88  |
| 3.12 | Aplicação de um limiar ( <i>threshold</i> ) para gerar a máscara de segmentação. . . . .                                                                                                                                                                                                 | 89  |
| 3.13 | Arquitetura completa da U-Net. . . . .                                                                                                                                                                                                                                                   | 91  |
| 3.14 | Arquitetura da U-Net multitarefa utilizada e as dimensões dos mapas de características ao longo da rede. . . . .                                                                                                                                                                         | 94  |
| 3.15 | Arquitetura da U-Net com a adição de uma camada Multitarefa e uma camada de inferência fuzzy. . . . .                                                                                                                                                                                    | 98  |
| 4.1  | Região do músculo <i>Longuissimus Dorsi</i> (em amarelo) e da garupa (em azul). . .                                                                                                                                                                                                      | 102 |
| 4.2  | Aquisição da imagem de ultrassom da região do olho de lombo. . . . .                                                                                                                                                                                                                     | 102 |
| 4.3  | Exemplo do processo de pré-processamento das imagens de ultrassom. . . . .                                                                                                                                                                                                               | 104 |
| 4.4  | Exemplo das máscaras de segmentação criadas para as imagens de ultrassom. . .                                                                                                                                                                                                            | 105 |

|      |                                                                                                                                                                                            |     |
|------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| 4.5  | Gráficos de custo, Coeficiente de Dice e acurácia ao longo das épocas. . . . .                                                                                                             | 110 |
| 4.6  | Gráficos de custo, Coeficiente de Dice e acurácia ao longo das épocas para o modelo de U-Net Fuzzy da AOL. . . . .                                                                         | 111 |
| 4.7  | Comparação entre as máscaras de segmentação da AOL. . . . .                                                                                                                                | 112 |
| 4.8  | Gráficos de custo, Coeficiente de Dice e acurácia ao longo das épocas para o modelo U-Net Multitarefa da EGL. . . . .                                                                      | 113 |
| 4.9  | Gráficos de custo, Coeficiente de Dice e acurácia ao longo das épocas para o modelo U-Net Fuzzy da EGL. . . . .                                                                            | 114 |
| 4.10 | Comparação entre as máscaras de segmentação. . . . .                                                                                                                                       | 115 |
| 4.11 | Gráficos de custo, Coeficiente de Dice e acurácia ao longo das épocas para os modelos U-Net Multitarefa da EGG. . . . .                                                                    | 116 |
| 4.12 | Gráficos de custo, Coeficiente de Dice e acurácia ao longo das épocas para os modelos U-Net Fuzzy da EGG. . . . .                                                                          | 117 |
| 4.13 | Comparação entre as máscaras de segmentação para a EGG. . . . .                                                                                                                            | 119 |
| 4.14 | Processo de extração de características e geração do mapa de características na primeira etapa de convolução da U-Net. . . . .                                                             | 120 |
| 4.15 | Mapa de características na última etapa de convolução do <i>bottleneck</i> da U-Net. .                                                                                                     | 121 |
| 4.16 | Mapa de características gerado com o módulo de inferência fuzzy e mapa de características gerado sem o módulo de inferência fuzzy. . . . .                                                 | 122 |
| 4.17 | Mapa de características gerado no terceiro bloco do <i>decoder</i> da U-Net Fuzzy e da U-Net Multitarefa. . . . .                                                                          | 123 |
| 4.18 | Tela inicial do aplicativo para seleção do indicador e modelo. . . . .                                                                                                                     | 124 |
| 4.19 | Tela de seleção das imagens de ultrassom a serem processadas. . . . .                                                                                                                      | 125 |
| 4.20 | Tela de visualização dos resultados, com as máscaras de segmentação sobrepostas à imagem original e os valores numéricos do indicador. . . . .                                             | 126 |
| 4.21 | Exemplo de uma imagem de ultrassom com baixa qualidade, em que a rede gerou uma segmentação errônea, e o aplicativo calculou apenas a região maior, eliminando as regiões menores. . . . . | 127 |

# Lista de Tabelas

|     |                                                                                                                         |     |
|-----|-------------------------------------------------------------------------------------------------------------------------|-----|
| 1   | Estudos selecionados sobre a utilização de U-Net na segmentação de imagens de ultrassom. . . . .                        | 24  |
| 1.1 | Banco de dados com as avaliações dos colegas e a decisão final de compra. . . . .                                       | 31  |
| 2.1 | Operações entre conjuntos clássicos. . . . .                                                                            | 68  |
| 4.1 | Quantidade de modelos desenvolvidos para cada medida. . . . .                                                           | 106 |
| 4.2 | Resumo da configuração de treinamento adotada nos experimentos. . . . .                                                 | 108 |
| 4.3 | Resumo dos resultados obtidos para a AOL, comparando o modelo de U-Net Multitarefa com o modelo de U-Net Fuzzy. . . . . | 112 |
| 4.4 | Resumo dos resultados obtidos para a EGL, comparando o modelo U-Net Multitarefa com o modelo U-Net Fuzzy. . . . .       | 115 |
| 4.5 | Resultados das métricas para a EGG. . . . .                                                                             | 118 |

# Sumário

|                                                                               |           |
|-------------------------------------------------------------------------------|-----------|
| Dedicatória                                                                   | 7         |
| Agradecimentos                                                                | 8         |
| Resumo                                                                        | 9         |
| Abstract                                                                      | 10        |
| Lista de Figuras                                                              | 13        |
| Lista de Tabelas                                                              | 14        |
| Introdução                                                                    | 18        |
| <b>1 Redes Neurais</b>                                                        | <b>26</b> |
| 1.1 Neurônio Biológico e Artificial . . . . .                                 | 26        |
| 1.1.1 Neurônio Biológico . . . . .                                            | 26        |
| 1.1.2 Unidade . . . . .                                                       | 27        |
| 1.1.3 Tipos de Função de Ativação . . . . .                                   | 29        |
| 1.1.4 <i>Perceptron</i> e Algoritmo de Aprendizagem . . . . .                 | 30        |
| 1.2 Redes Neurais . . . . .                                                   | 35        |
| 1.2.1 <i>Backpropagation</i> . . . . .                                        | 37        |
| 1.2.2 Gradiente Descendente Estocástico . . . . .                             | 39        |
| 1.2.3 <i>Overfitting</i> . . . . .                                            | 40        |
| 1.2.4 Hiperparâmetros . . . . .                                               | 40        |
| 1.2.5 Regularização . . . . .                                                 | 41        |
| 1.2.6 <i>Momentum</i> e Adam . . . . .                                        | 43        |
| 1.2.7 <i>Vanishing Gradient</i> . . . . .                                     | 44        |
| 1.2.8 Função de Custo <i>Binary Cross-Entropy</i> . . . . .                   | 45        |
| 1.2.9 Arquiteturas de Redes Neurais . . . . .                                 | 46        |
| 1.2.10 Aplicação da Arquitetura MLP para Classificação de Dígitos Manuscritos | 47        |
| 1.2.11 Aplicação da Arquitetura MLP para Reconstrução de Figura Geométrica    | 50        |
| 1.3 Rede Neural para Análise de Imagens . . . . .                             | 51        |
| 1.3.1 Córtex Visual Humano e Visão Computacional . . . . .                    | 51        |

|          |                                                                    |            |
|----------|--------------------------------------------------------------------|------------|
| 1.3.2    | Redes Neurais Convolucionais . . . . .                             | 53         |
| 1.3.3    | Convolução . . . . .                                               | 53         |
| 1.3.4    | Correlação Cruzada . . . . .                                       | 60         |
| 1.3.5    | <i>Kernel</i> Multicanais . . . . .                                | 60         |
| 1.3.6    | <i>Padding</i> . . . . .                                           | 60         |
| 1.3.7    | <i>Stride</i> . . . . .                                            | 61         |
| 1.3.8    | <i>Pooling</i> . . . . .                                           | 62         |
| 1.3.9    | Convolução Transposta . . . . .                                    | 63         |
| <b>2</b> | <b>Teoria dos Conjuntos Fuzzy</b>                                  | <b>66</b>  |
| 2.1      | Conjuntos Fuzzy . . . . .                                          | 66         |
| 2.1.1    | Operações entre Conjuntos Fuzzy . . . . .                          | 67         |
| 2.1.2    | Normas Triangulares . . . . .                                      | 69         |
| 2.1.3    | Níveis de um Conjunto Fuzzy . . . . .                              | 70         |
| 2.1.4    | Números Fuzzy . . . . .                                            | 71         |
| 2.2      | Sistema Baseado em Regras Fuzzy . . . . .                          | 73         |
| 2.2.1    | Regras e Inferência Fuzzy . . . . .                                | 73         |
| 2.2.2    | Componentes de um SBRF . . . . .                                   | 74         |
| 2.3      | ANFIS . . . . .                                                    | 78         |
| 2.3.1    | Treinamento . . . . .                                              | 80         |
| <b>3</b> | <b>Estrutura das Redes Neurais U-Net e U-Net Fuzzy</b>             | <b>82</b>  |
| 3.1      | Noções Básicas da Arquitetura U-Net . . . . .                      | 82         |
| 3.1.1    | Arquitetura Original da U-Net . . . . .                            | 90         |
| 3.2      | Modelos Propostos . . . . .                                        | 91         |
| 3.2.1    | Visão Geral das Modificações Propostas . . . . .                   | 92         |
| 3.3      | Critérios de Avaliação . . . . .                                   | 98         |
| 3.3.1    | Coeficiente de Dice . . . . .                                      | 98         |
| 3.3.2    | Acurácia . . . . .                                                 | 99         |
| 3.3.3    | Erro Médio Absoluto . . . . .                                      | 100        |
| <b>4</b> | <b>Aplicação da U-Net e U-Net Fuzzy em Ultrassonografia Bovina</b> | <b>101</b> |
| 4.1      | Base de Dados e Planejamento Experimental . . . . .                | 101        |
| 4.1.1    | Contexto do Problema e Medidas Avaliadas . . . . .                 | 101        |
| 4.1.2    | Conjunto de Dados . . . . .                                        | 103        |
| 4.1.3    | Pré-processamento e Máscaras de Segmentação . . . . .              | 103        |
| 4.2      | Configuração dos Modelos . . . . .                                 | 106        |
| 4.2.1    | Modelos Comparados . . . . .                                       | 106        |
| 4.2.2    | Configuração de Treinamento . . . . .                              | 107        |
| 4.2.3    | Métricas de Avaliação . . . . .                                    | 108        |
| 4.2.4    | Ambiente Computacional . . . . .                                   | 108        |
| 4.3      | Resultados e Análise . . . . .                                     | 109        |

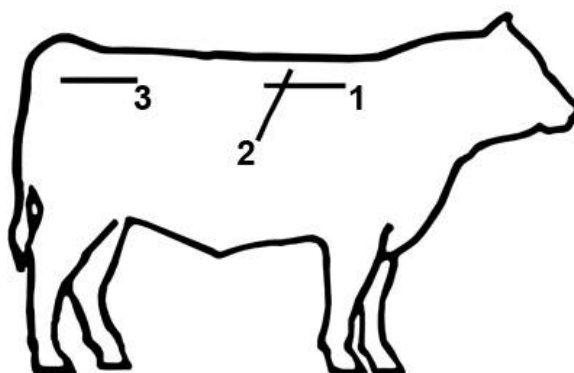
|          |                                        |            |
|----------|----------------------------------------|------------|
| 4.3.1    | AOL . . . . .                          | 109        |
| 4.3.2    | EGL . . . . .                          | 112        |
| 4.3.3    | EGG . . . . .                          | 116        |
| 4.4      | Análise Geral dos Resultados . . . . . | 119        |
| 4.5      | Aplicativo Computacional . . . . .     | 124        |
| <b>5</b> | <b>Considerações Finais</b>            | <b>128</b> |
|          | <b>Referências Bibliográficas</b>      | <b>131</b> |
|          | <b>ANEXO A - Comissão de Ética</b>     | <b>137</b> |

# Introdução

O desenvolvimento econômico e social do Brasil está profundamente enraizado em atividades agrícolas e pecuárias (GARCAO, 2015). De acordo com a Associação Brasileira das Indústrias Exportadoras de Carnes (ABIEC), o complexo agroindustrial voltado à produção de carne bovina representa cerca de 8,4% do Produto Interno Bruto (PIB) nacional (ABIEC, 2021). Além disso, o Brasil possui o maior rebanho bovino comercial do mundo. Em 2021, o país concentrava 11,6% do rebanho mundial e liderava o comércio internacional de carne bovina, sendo responsável por aproximadamente 21% das exportações globais (GUARALDO, 2021). Em abril de 2026, o Brasil foi responsável por aproximadamente 30% das exportações mundiais (UNITED STATES DEPARTMENT OF AGRICULTURE, 2026).

Nesse cenário, a avaliação precisa das características da carcaça dos animais desempenha papel crucial para a cadeia produtiva, pois permite determinar indicadores relacionados ao desenvolvimento dos músculos, acúmulo de gordura na carcaça e qualidade da carne (WILSON, 1998). Entre os métodos disponíveis para esta avaliação, a ultrassonografia se destaca por possibilitar o exame *in vivo*, de forma rápida, com boa precisão e custos relativamente baixos (SUGUISAWA; MATOS.; SUGUISAWA, 2013). Por meio de imagens de ultrassom, é possível mensurar características como a Área de Olho de Lombo (AOL) e a Espessura de Gordura subcutânea no Lombo (EGL), localizadas no músculo *longissimus dorsi* (Contra-filé) entre a 12ª e a 13ª costela e a Espessura de Gordura subcutânea na Garupa (EGG), medida no começo do músculo *Biceps Femolis* (ponta da picanha). Na Figura 1 é apresentada uma ilustração da localização das medidas no animal.

Figura 1: Ilustração da localização das regiões de interesse.

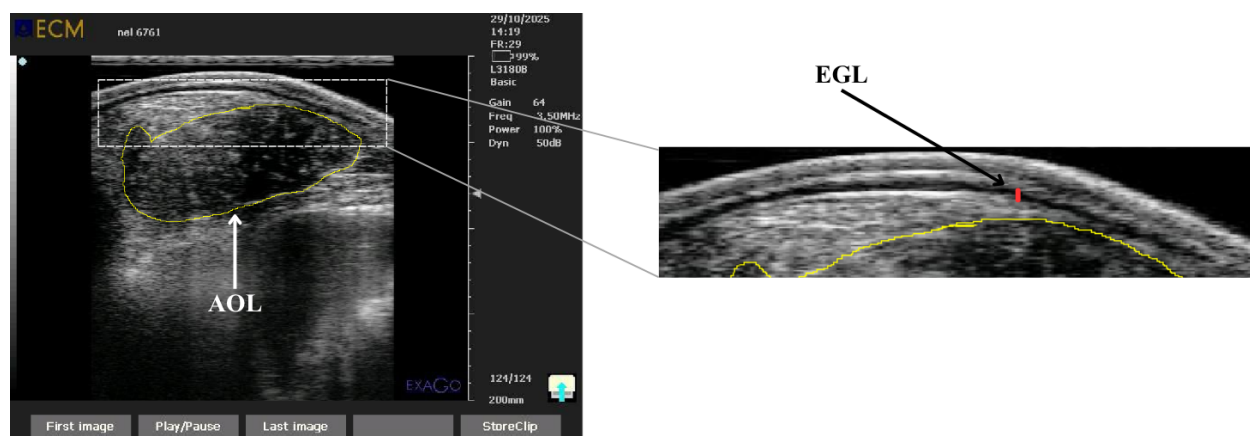


Fonte: Adaptado de Wilson (1998).

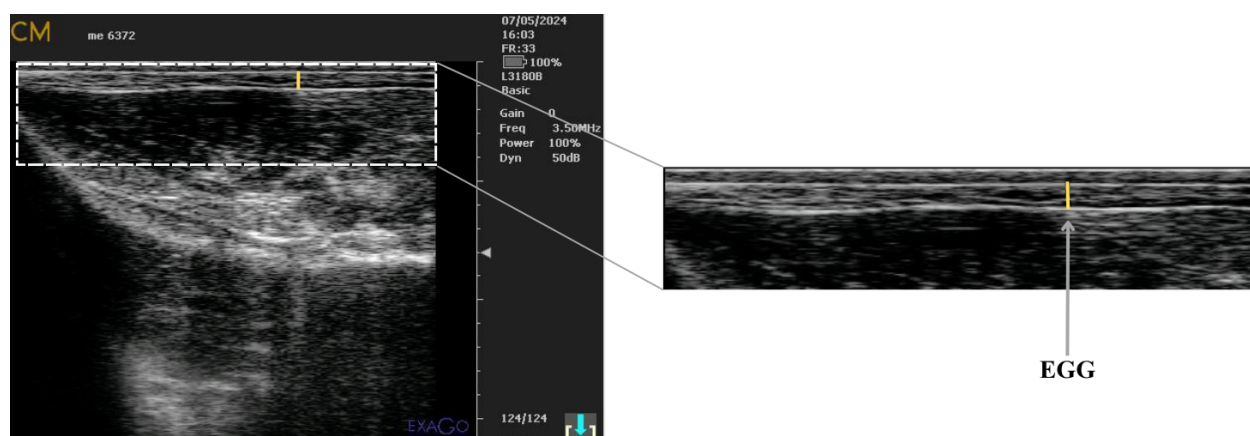
Na Figura 1, a linha horizontal (1) indica a localização da espessura de gordura subcutânea no lombo onde é possível medir a EGL, a linha inclinada (2) indica a localização do Olho do Lombo onde é possível medir a AOL e a linha horizontal (3) indica a região de medição da EGG.

As imagens de ultrassom correspondentes a essas regiões são apresentadas na Figura 2, as quais foram fornecidas pelo Centro Avançado de Pesquisa e Desenvolvimento de Bovinos de Corte, Instituto de Zootecnia, SAA-SA, localizado em Sertãozinho - SP.

Figura 2: Imagens de ultrassom das regiões de interesse.



(a) AOL e EGL.



(b) EGG.

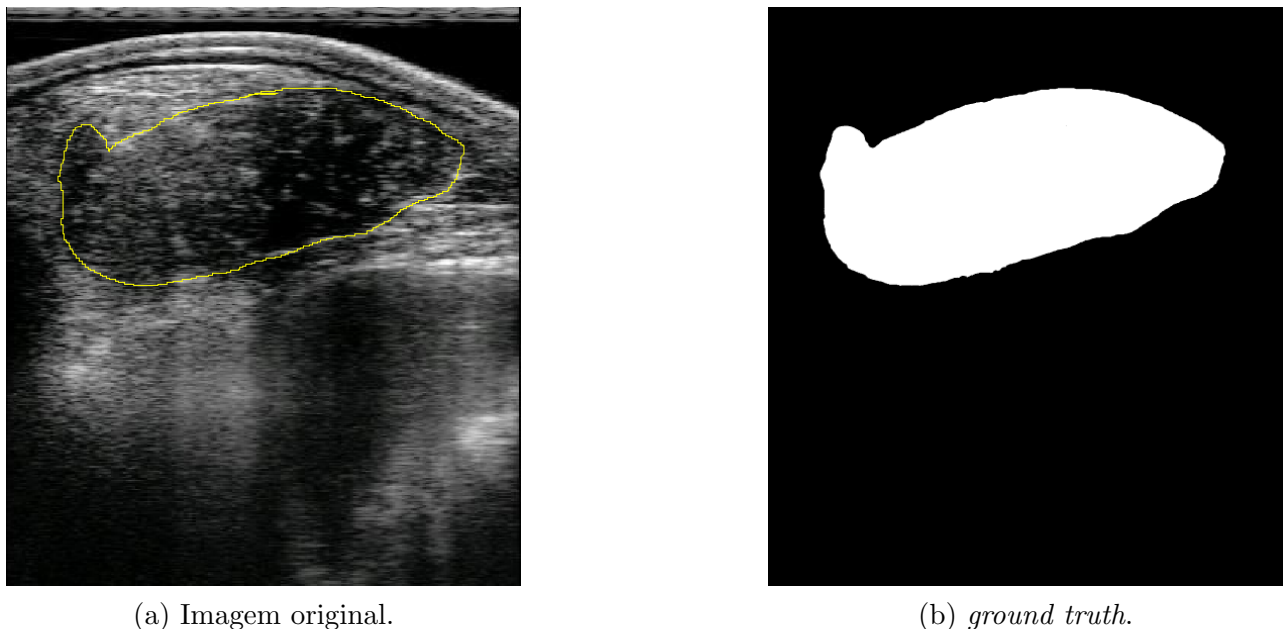
Fonte: Elaborado pelo autor.

A delimitação do contorno da região em amarelo na Figura 2(a) corresponde à AOL, na mesma imagem encontra-se a EGL, demarcada em vermelho. A linha amarela na Figura 2(b) corresponde à EGG. A variabilidade nessas características influencia a comercialização e os resultados econômicos dos produtores (FELICIO, 2010), tornando a mensuração precisa uma demanda constante do setor.

Tradicionalmente, a delimitação e mensuração das regiões são realizadas manualmente por especialistas, um processo sujeito à subjetividade e que demanda tempo considerável. Diante disso, a aplicação de técnicas de visão computacional e inteligência artificial (SZELISKI, 2010) apresenta-se como uma alternativa promissora para automatizar e padronizar esse pro-

cesso. Em particular, arquiteturas de redes neurais convolucionais, em inglês *Convolutional Neural Networks* (CNNs) (O'SHEA; NASH, 2015), uma subárea da inteligência artificial dedicada à análise e interpretação de imagens, têm demonstrado resultados expressivos em diversas aplicações biomédicas, tal como, a análise de estruturas neuronais (RONNEBERGER; FISCHER; BROX, 2015). Uma das principais técnicas empregadas nestas redes é a segmentação de imagens, que consiste na classificação de cada pixel em uma categoria específica, possibilitando a identificação precisa das regiões de interesse. Para o treinamento e a avaliação desses modelos, utiliza-se uma máscara de segmentação, comumente denominada *ground truth*, criada manualmente por especialistas. Essa máscara corresponde a uma imagem binária na qual os pixels pertencentes à região de interesse são destacados em branco, enquanto os demais são representados em preto. Na Figura 3, é apresentada uma imagem de ultrassom e o *ground truth* criado para mensurar a AOL.

Figura 3: Exemplo de *ground truth*.



Fonte: Elaborado pelo autor.

Na imagem original, Figura 3(a), a região de interesse está delimitada de amarelo, enquanto na Figura 3(b), pode-se visualizar o *ground truth* correspondente da Figura 3(a).

Entre as arquiteturas de redes neurais utilizadas para segmentação de imagens, destaca-se a U-Net. Seu nome deriva do formato de sua estrutura, que se assemelha à letra “U” (RONNEBERGER; FISCHER; BROX, 2015). Este modelo reduz a resolução da imagem recebida extraindo características, desde detalhes locais até padrões gerais, etapa conhecida como *encoder*, e, em seguida, uma representação é construída a partir dessas características extraídas, etapa conhecida como *decoder*. Este processo permite a geração de uma máscara de segmentação predita pelo modelo. Essa máscara é posteriormente comparada ao *ground truth*, utilizando métricas de similaridade, como o coeficiente de Dice, cuja descrição é apresentada adiante.

A U-Net foi originalmente proposta para segmentação de imagens biomédicas, sendo re-

conhecida pela capacidade de operar com conjuntos de dados reduzidos e pela eficiência na captura de informações em imagens. Quando é necessário que a arquitetura faça algo além da máscara de segmentação, como por exemplo, classificação, utiliza-se uma abordagem denominada Multitarefa, do inglês *Multitask*. Essa abordagem permite que a rede neural aprenda a realizar múltiplas tarefas simultaneamente.

Paralelamente, a teoria dos conjuntos fuzzy (ZADEH, 1965) oferece ferramentas para o tratamento de incertezas e imprecisões inerentes a imagens de ultrassom, onde as fronteiras entre as estruturas anatômicas nem sempre são claramente definidas. A integração de redes neurais com a teoria dos conjuntos fuzzy (PURWONO *et al.*, 2025), por meio do Sistema de Inferência Neuro-Fuzzy Adaptativo, do inglês *Adaptive Neuro-Fuzzy Inference System* (ANFIS), possibilita combinar a capacidade de aprendizado das redes neurais com a flexibilidade de um Sistema Baseado em Regras Fuzzy (SBRF) no tratamento de informações imprecisas. Alguns estudos têm explorado a aplicação da U-Net em tarefas de segmentação de imagens de ultrassom da região de olho de lombo (MELO *et al.*, 2022), como também a junção da U-Net com integração ao ANFIS para reconhecimento de imagens microscópicas de células (KIRICHEV; SLAVOV; MOMCHEVA, 2021).

Neste contexto, este trabalho tem como objetivo geral:

- Desenvolver e comparar os modelos U-Net com abordagem Multitarefa e o modelo U-Net Fuzzy, que é uma U-Net com abordagem Multitarefa integrada ao ANFIS, para a delimitação automática das regiões de interesse e mensuração dos indicadores AOL, EGL e EGG em imagens de ultrassom de carcaça de bovinos das raças Nelore e Caracu.

Para alcançar o objetivo geral, são estabelecidos os seguintes objetivos específicos:

- Compreender os conceitos fundamentais de redes neurais, teoria dos conjuntos fuzzy, sistemas baseados em regras fuzzy e apresentar a arquitetura U-Net Multitarefa e sua integração com o ANFIS para segmentação de imagens;
- Padronizar o banco de dados das imagens de ultrassom e desenvolver as respectivas máscaras de segmentação;
- Aplicar técnicas de pré-processamento e aumento de dados para adequar o banco de dados ao treinamento dos modelos;
- Desenvolver e treinar os modelos propostos utilizando as bibliotecas de aprendizagem de máquina disponíveis na linguagem de programação Python versão 3.11.15 (PYTHON SOFTWARE FOUNDATION, 1990), TensorFlow versão 2.21.0 com suporte a GPU (GOOGLE LLC, 2015) e Keras versão 3.13.2 (KERAS TEAM, 2015);
- Produzir um aplicativo computacional em Python, utilizando a biblioteca *Tkinter* para a implementação da interface gráfica, a fim de automatizar a delimitação das regiões de interesse e a mensuração dos indicadores AOL, EGL e EGG em imagens de ultrassom (PYTHON SOFTWARE FOUNDATION, 1991).

A comparação entre os modelos visa avaliar a eficácia de cada um na predição das máscaras de segmentação e com isso na obtenção das medidas de AOL, EGL e EGG, contribuindo para a automação do processo de avaliação de características da carcaça em bovinos.

No trabalho são utilizadas 135 imagens de ultrassom bovinas, das quais 108 são da raça Nelore e 27 da raça Caracu, todos machos com idade média de 12 meses, não sendo necessário passar pelo comitê de ética para a realização do estudo, conforme aprovado pelo ANEXO A - Comissão de Ética (5). As imagens são obtidas por meio do aparelho de ultrassom Aloka SSD-500, aparelho equipado com uma sonda linear de 3,5 MHz (Aloka Co. Ltd., 1990), utilizado por um especialista. Por conta do número limitado de imagens disponíveis, foi necessário realizar um processo de aumento de dados, do inglês *data augmentation*, para ampliar a quantidade de imagens. O processo de aumento de dados incluiu técnicas de rotação e adição de ruído às imagens originais por meio do aumento e da diminuição do brilho, simulando variações que podem ocorrer durante a aquisição das imagens de ultrassom. Para avaliar o desempenho dos modelos propostos, utilizam-se três métricas:

- O coeficiente de Dice, que quantifica a similaridade entre a máscara de segmentação desenvolvida pelo especialista (*ground truth*) e a máscara de segmentação predita pelo modelo (DICE, 1945).
- A Acurácia, que avalia a capacidade do modelo em classificar corretamente as regiões de interesse comparando com o *ground truth* (MELO *et al.*, 2022).
- Erro Médio Absoluto, do inglês *Mean Absolute Error* (MAE), compara as medidas obtidas pelo especialista com as medidas extraídas da máscara de segmentação predita pelo modelo (MELO *et al.*, 2022).

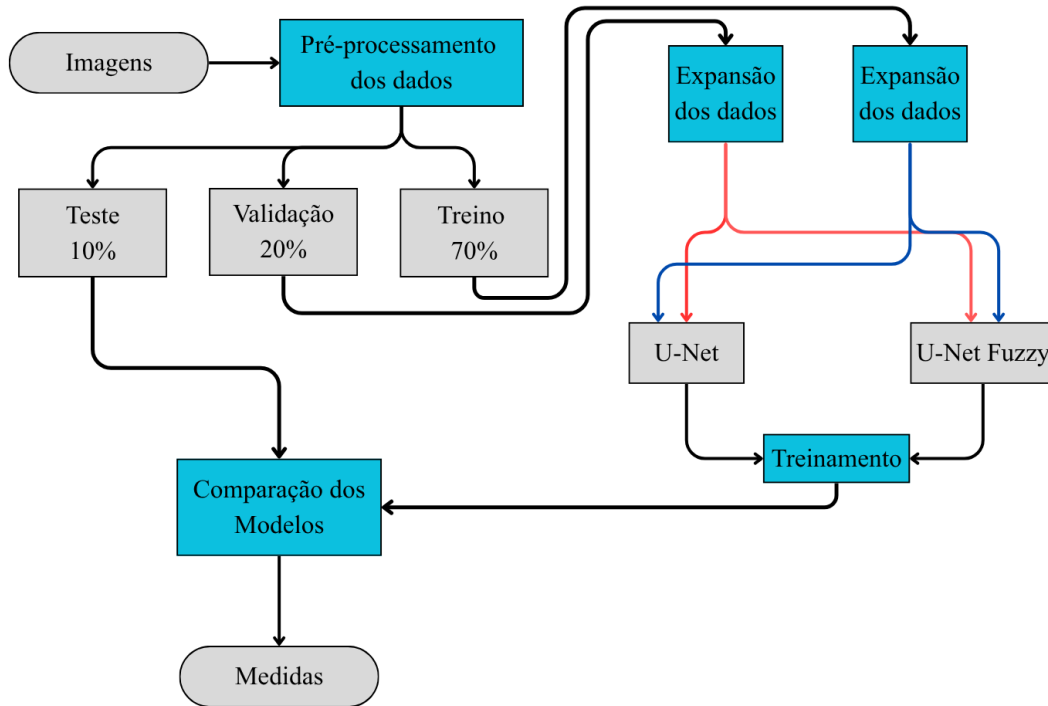
Para cada um dos indicadores mensurados AOL, EGL e EGG, são desenvolvidos dois modelos, denominados U-Net Multitarefa e U-Net Fuzzy. Na Figura 4 é apresentado o fluxo de treinamento dos modelos desenvolvidos, assim como a divisão do banco de dados em conjuntos de treinamento (70%), validação (20%) e teste (10%).

O fluxo apresentado na Figura 4 é repetido quatro vezes de forma aleatória para cada modelo. Este procedimento é feito para verificar a consistência dos resultados, ou seja, se os resultados não apresentam variações significativas, com diferentes particionamentos do conjunto de dados.

No processo de treinamento dos modelos, o banco de imagens é pré-processado e dividido em conjuntos de treinamento, validação e teste. Após a divisão, os conjuntos de treinamento e validação são expandidos, ou seja, *data augmentation*, e utilizados para o treinamento dos modelos, enquanto o conjunto de teste é reservado para a avaliação final do desempenho dos modelos.

Por fim, foi desenvolvido um aplicativo computacional utilizando a biblioteca disponível na linguagem de programação Python, *Tkinter*, que faz a implementação de uma interface gráfica para os modelos desenvolvidos, automatizando a delimitação da região e a mensuração dos indicadores AOL, EGL e EGG nas imagens de ultrassom, que poderá ser disponibilizado

Figura 4: Fluxo de treinamento dos modelos.



Fonte: Elaborado pelo autor.

para o Centro Avançado de Pesquisa e Desenvolvimento de Bovinos de Corte, o que torna a tecnologia acessível para produtores e profissionais do setor agropecuário, podendo contribuir para a melhoria da eficiência e precisão na avaliação das características da carcaça dos bovinos. Até o momento, não foram encontradas, na literatura consultada, contribuições científicas que proponham modelos capazes de segmentar imagens de ultrassom por meio de uma arquitetura U-Net integrada ao ANFIS (U-Net Fuzzy), permitindo a mensuração dos indicadores AOL, EGL e EGG. Consequentemente, não foram encontrados trabalhos que comparem a U-Net Multitarefa com a U-Net Fuzzy para esses indicadores.

Na Tabela 1 é sintetizado os estudos selecionados que empregam a U-Net ou variantes para segmentação em imagens de ultrassom. O único trabalho encontrado sobre o tema de segmentação de imagens de ultrassom bovinas comparou a U-Net com outras arquiteturas de redes neurais, exclusivamente para a AOL (MELO *et al.*, 2022). Os trabalhos (AMIRI *et al.*, 2020) e (SZENTIMREY *et al.*, 2022), utilizam as métricas de Distância de Hausdorff e Intersection over Union (IoU), em português, interseção sobre união. A primeira métrica consiste em medir o erro máximo entre as fronteiras das regiões segmentadas e as regiões de referência, enquanto a segunda mede a sobreposição entre as regiões segmentadas e as regiões de referência. Os trabalhos mencionados na Tabela 1 utilizam a arquitetura da U-Net como base e implementam modificações para melhorar o desempenho em suas respectivas aplicações.

Tabela 1: Estudos selecionados sobre a utilização de U-Net na segmentação de imagens de ultrassom.

| Estudo                                   | Aplicação                                       | Proposta baseada em U-Net                                                                      | Evidência reportada                                                                                               |
|------------------------------------------|-------------------------------------------------|------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------|
| Amiri <i>et al.</i> (2020)               | Lesões mamárias em ultrassom 2D                 | Segmentação em dois estágios com U-Net e <i>test-time augmentation</i> .                       | Avaliação por Dice, IoU e distância de Hausdorff; o aumento no teste foi usado para refinar a máscara predita.    |
| Guo <i>et al.</i> (2021)                 | Tumores mamários em ultrassom 2D                | U-Net expandida, com <i>dropout</i> e treinamento expandido para segmentação e reconhecimento. | Dice = 90,5% e IoU = 82,7%; Dice 7,6 pontos percentuais acima da U-Net convencional.                              |
| Melo <i>et al.</i> (2022)                | Carcça bovina: área de olho de lombo            | U-Net++ para delimitação automática da área de olho de lombo.                                  | Acurácia de 97,53%; aplicação mais próxima deste trabalho, porém restrita à AOL.                                  |
| Szentimrey <i>et al.</i> (2022)          | Ventrículos cerebrais neonatais em ultrassom 3D | Conjunto de três variantes de U-Net 3D para segmentação volumétrica.                           | Avaliação por Dice, erro de volume e distância de Hausdorff, demonstrando a extensão da arquitetura a volumes 3D. |
| Zheng <i>et al.</i> (2023)               | Tireoide e nódulos em ultrassom 2D              | DSRU-Net: U-Net com blocos ResNeSt, ASPP e convolução deformável.                              | Dice médio = 92,5% e Dice dos nódulos = 94,1%, superiores aos da U-Net de referência.                             |
| Pramanik <i>et al.</i> (2023)            | Tumores mamários em ultrassom 2D                | DBU-Net com dois codificadores: imagem original e bordas obtidas pelo operador de Roberts.     | Dice = 85,28% (BUSI) e 87,28% (UDIAT).                                                                            |
| Malekmohammadi, Soryani e Kozegar (2024) | Massas mamárias em ultrassom automatizado 3D    | U-Net atencional aprimorada para segmentação de massas em ABUS.                                | Compara variantes de U-Net em dados ABUS por métricas de sobreposição, com foco em massas de mama.                |

Neste trabalho, adotou-se o ponto como separador decimal nos eixos de gráficos, pois os sistemas computacionais utilizados estão configurados em inglês.

O presente trabalho está organizado da seguinte forma. No Capítulo 1, são apresentados os conceitos fundamentais de redes neurais, desde a inspiração biológica do neurônio artificial até as arquiteturas de redes neurais convolucionais. No Capítulo 2 é introduzida a teoria dos conjuntos fuzzy e os sistemas baseados em regras fuzzy, que fornecem a base teórica para a construção do ANFIS. No Capítulo 3 é descrito detalhadamente a implementação dos modelos propostos, incluindo a arquitetura da U-Net Multitarefa e a integração do ANFIS. No Capítulo 4 são apresentados a organização do conjunto de dados, o processo de treinamento e os resultados

obtidos, com uma análise comparativa entre os modelos e o aplicativo desenvolvido. Por fim, no Capítulo 5 é apresentada as considerações finais, destacando as contribuições e sugestões para pesquisas futuras.

# Capítulo 1

## Redes Neurais

A ideia de máquinas inteligentes desperta fascínio e receio há décadas. No imaginário popular, esse medo costuma aparecer em obras de ficção científica, como *O Exterminador do Futuro*, em que sistemas artificiais ultrapassam o controle humano e passam a agir de forma autônoma. Embora esse cenário ainda pertença ao campo da ficção, revela uma inquietação real: até que ponto uma máquina pode aprender, decidir e interpretar o mundo? Na prática, os modelos atuais estão muito distantes de uma inteligência artificial consciente, mas representam o estágio mais avançado já alcançado pelo ser humano na construção de sistemas capazes de reconhecer padrões, extrair informações e auxiliar na resolução de problemas complexos. Nesse contexto, compreender os fundamentos das redes neurais torna-se essencial para entender como essas tecnologias funcionam e por que elas têm alcançado resultados tão expressivos em tarefas de geração de texto, classificação de objetos e análise de imagens.

### 1.1 Neurônio Biológico e Artificial

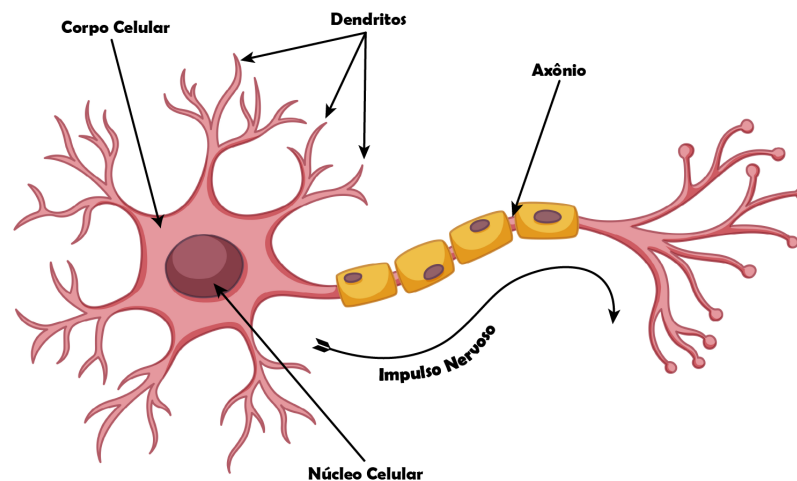
Na área de Inteligência Artificial, comumente utiliza-se o termo *neurônio artificial* para se referir a uma unidade de processamento em um modelo de rede neural. O neurônio artificial é inspirado no funcionamento do neurônio biológico, que é a unidade básica do sistema nervoso. Assim como os neurônios biológicos, os artificiais recebem entradas, processam essas entradas e produzem uma saída. A principal função de um neurônio artificial é realizar uma combinação linear das entradas, seguida por uma função de ativação não linear, que determina a saída do neurônio. A semelhança entre esses dois objetos (Neurônio artificial/Neurônio Biológico) só está na inspiração, sendo bem diferentes em seu funcionamento, com isso, neste trabalho utiliza-se o termo *unidade* para se referir ao neurônio artificial (O'SHEA; NASH, 2015). Os conceitos apresentados a seguir são fundamentais para o entendimento do funcionamento das redes neurais, que são abordadas na Seção 1.2.

#### 1.1.1 Neurônio Biológico

Para entender o funcionamento da unidade é necessário entender o funcionamento do neurônio biológico. Neurônios são “células excitáveis” que se comunicam ao transmitir impulsos elétricos.

Células *excitáveis* são definidas como capazes de produzir sinais elétricos amplos e rápidos denominados *potenciais de ação* (STANFIELD, 2014). A maioria dos neurônios contém três componentes principais: um corpo celular, o(s) dendrito(s) e um axônio. O corpo celular, chamado de *soma* contém o núcleo celular, local onde parte da informação recebida de outras partes do corpo é manipulada. Os *dendritos* se ramificam a partir do corpo celular e são responsáveis por receber e integrar uma imensa quantidade de sinais provenientes de outros neurônios. Outra ramificação advinda do corpo celular é o *axônio*. Diferente dos dendritos, cuja função é *receber* informação, um axônio *envia* informações. Grande parte dos neurônios tem apenas um axônio, mas os axônios conseguem se ramificar, enviando sinais a mais de um destino. Esses três componentes, de forma resumida, formam o neurônio biológico, como mostrado na Figura 1.1.

Figura 1.1: Estrutura de um neurônio típico.



Fonte: Adaptado de Stanfield (2014).

Outro fator importante em um neurônio biológico são os potenciais de ação, gerados nos dendritos, são avaliados e ponderados por meio da *força sináptica*, que diz o quão importante é a informação (BEAR; CONNORS; PARADISO, 2017). Após esse processo, os potenciais de ação são enviados para o núcleo celular. Entre o corpo celular e o axônio é decidido se o potencial é excitatório ou inibitório. A decisão é feita caso o potencial de ação supere ou não um *limiar*. Caso seja excitatório, a informação é passada a diante, caso seja inibitório, a informação é suprimida.

### 1.1.2 Unidade

A partir das informações obtidas com o estudo do neurônio biológico, pesquisadores tentaram simular seu funcionamento em computador. A implementação mais bem aceita de simulação foi proposta por dois pesquisadores, o neurobiologista Warren McCulloch e o matemático Walter Pitts em 1943 que utilizaram da lógica proposicional para descrever o funcionamento do

sistema nervoso central (MCCULLOCH; PITTS, 1943). McCulloch acreditava que o neurônio biológico tinha uma natureza binária, ou seja, um neurônio influencia outro ou não, entretanto, posteriormente essa crença mostrou-se errônea com os avanços de estudos na área.

Para entender o funcionamento da unidade proposta por McCulloch e Pitts, considerando que as variáveis  $x_1, x_2, x_3, \dots, x_n$  são denominadas *dados de entrada*, em analogia às sinapses de um neurônio biológico. Os valores  $w_1, w_2, w_3, \dots, w_n$  são os *pesos* associados a cada dado de entrada. O viés tem a mesma funcionalidade do limiar no neurônio biológico, determinando se o neurônio é ativado ou não. Para resumir a notação, defini-se o viés de  $b$ , do inglês *bias*. Na soma, a variável  $z$  é obtida fazendo a combinação linear das variáveis:

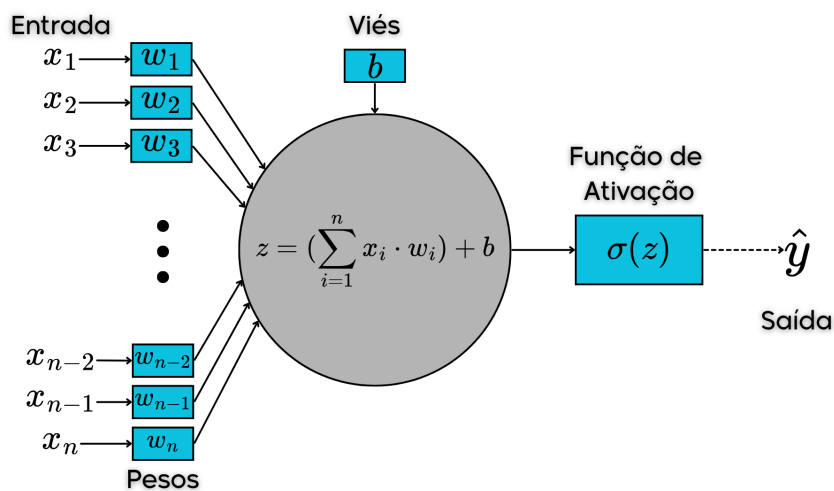
$$x_1 \cdot w_1 + x_2 \cdot w_2 + x_3 \cdot w_3 + \dots + x_n \cdot w_n + b = z = \sum_{i=1}^n x_i \cdot w_i + b.$$

Em seguida, o valor  $z$  é aplicado a uma *função de ativação*  $\sigma : \mathbb{R} \rightarrow \mathbb{R}$ , resultando na saída  $\hat{y}$ , ou seja:

$$\hat{y} = \sigma \left( \sum_{i=1}^n w_i \cdot x_i + b \right). \quad (1.1)$$

Na Figura 1.2 é possível visualizar o modelo de uma unidade de McCulloch e Pitts, em que  $x_1, x_2, x_3, \dots, x_n$  são os dados de entrada,  $w_1, w_2, w_3, \dots, w_n$  são os pesos associados a cada dado de entrada e  $b$  é o viés. O valor  $z$  é a combinação linear das variáveis e  $\hat{y}$  é a saída da unidade após a aplicação da função de ativação  $\sigma$ .

Figura 1.2: Modelo de uma unidade de McCulloch e Pitts.



Fonte: Elaborado pelo autor.

Na unidade de McCulloch e Pitts, a função de ativação é uma função de limiar, a unidade é ativada, ou seja, produz uma saída de 1, se a soma ponderada das entradas mais o viés for maior ou igual a zero, caso contrário, a saída é 0. Os dados de entrada são valores binários,  $x_i \in \{0, 1\}$  para  $i = 1, 2, \dots, n$  e os pesos e viés são definidos manualmente. O modelo de McCulloch e Pitts é capaz de representar funções lógicas como *AND*, *OR* e *NOT*. Por exemplo, para representar

a função lógica *AND*, os pesos podem ser definidos como  $w_1 = w_2 = \dots = w_n = 1$  e o viés pode ser definido como  $b = -n + 0,5$ , em que  $n$  é o número de entradas. Com essa configuração, a unidade produzirá uma saída de 1 apenas quando todas as entradas forem 1, caso contrário, a saída é 0.

### 1.1.3 Tipos de Função de Ativação

A função de ativação, representada por  $\sigma$ , define a saída de uma unidade, pode-se definir diferentes tipos de funções de ativação, dependendo do comportamento desejado para a unidade. As funções de ativação mais comuns incluem:

- **Função de Limiar:** Para este tipo de função de ativação, descrito na equação (1.2), a unidade é ativada, ou seja, produz uma saída de 1, se a soma ponderada das entradas mais o viés for maior ou igual a zero. Caso contrário, a saída é 0, definida por:

$$\sigma(z) = \begin{cases} 1, & \text{se } z \geq 0 \\ 0, & \text{se } z < 0 \end{cases}. \quad (1.2)$$

- **Função Sigmoidal:** A função sigmoidal, cujo gráfico tem a forma de um “S”, é definida como uma função estritamente crescente que exhibe um balanceamento adequado entre comportamento linear e não linear. Um exemplo de função sigmoidal é a *sigmoid* logística, que foi amplamente usada no início das pesquisas em redes neurais pois a resposta do neurônio biológico é aproximadamente similar à função sigmoidal. Os estudos realizados por Lettvin *et al.* (1959) mostraram que os neurônios dos olhos de sapos respondem a estímulos visuais de maneira não linear, o que é consistente com o comportamento de uma função sigmoidal. O principal motivo para o seu desuso foi a dificuldade de treinamento em redes neurais profundas (ver Seção 1.2) devido ao problema do *vanish gradient* (ver Subseção 1.2.7). A função *sigmoid* logística é definida por:

$$\sigma(z) = \frac{1}{1 + \exp(-\alpha \cdot z)}, \quad (1.3)$$

em que  $\alpha$  é um parâmetro que controla a inclinação da curva. A função sigmoid logística tem um conjunto de propriedades que se mostram úteis nos cálculos relacionados à aprendizagem dos pesos:

1. Não linear;
2. Contínua e diferenciável em todo o domínio de  $\mathbb{R}$ ;
3. A derivada tem forma simples e é expressa pela própria função:

$$\sigma'(z) = \sigma(z) \cdot (1 - \sigma(z));$$

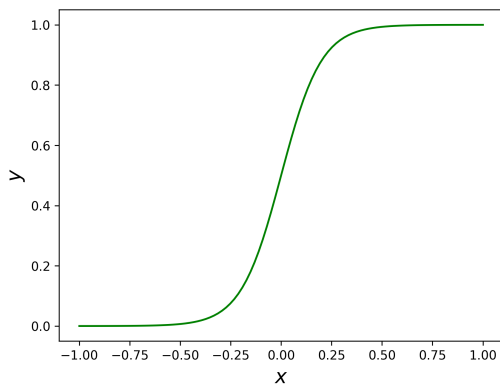
4. É estritamente monótona, ou seja,  $z_1 \leq z_2 \iff \sigma(z_1) \leq \sigma(z_2)$ .

- Função *Rectified Linear Unit* (ReLU): A função ReLU, em português, Unidade Linear Retificada, é definida como  $\sigma(z) = \max(0, z)$ . Esta é amplamente utilizada em redes neurais profundas, estudadas na Seção 1.2, devido à sua simplicidade computacional. A função ReLU ativa a unidade apenas quando a soma ponderada das entradas mais o viés é positiva, caso contrário, a saída é zero, ou seja:

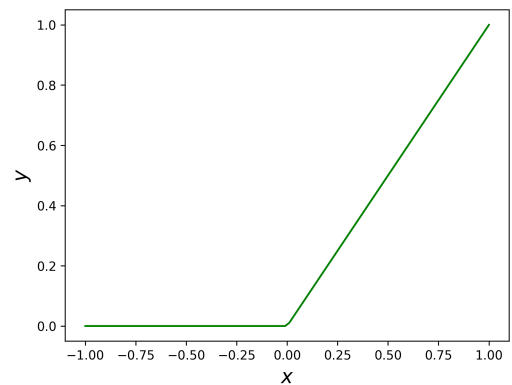
$$\sigma(z) = \begin{cases} z, & \text{se } z > 0 \\ 0, & \text{se } z \leq 0 \end{cases}. \quad (1.4)$$

Em geral, o objetivo da função de ativação é limitar a saída de uma unidade para um intervalo de valores. Na Figura 1.3(a) é mostrado o gráfico da função *sigmoid* logística, e na Figura 1.3(b) é mostrado o gráfico da função *Rectified Linear Unit*.

Figura 1.3: Gráficos das funções de ativação.



(a) Função *sigmoid*.



(b) Função ReLU.

Fonte: Elaborado pelo autor.

A principal desvantagem na unidade de McCulloch e Pitts é a ausência de um algoritmo de aprendizagem, ou seja, uma maneira de ajustar os pesos e o viés da unidade para que este seja capaz de resolver um problema específico. Essa limitação foi superada em 1958 por Frank Rosenblatt, que propôs o *perceptron* (ROSENBLATT, 1958), um modelo de unidade com um algoritmo de aprendizagem capaz de ajustar os pesos e o viés da unidade.

#### 1.1.4 *Perceptron* e Algoritmo de Aprendizagem

O *perceptron* é um modelo de unidade que utiliza um algoritmo de aprendizagem supervisionada para ajustar seus pesos e viés com base em um conjunto de dados “rotulados”. Dados rotulados são conjuntos de dados de entrada associados a saídas desejadas, ou seja, para cada elemento do conjunto de entradas, há uma saída correta conhecida definida por algum especialista. O objetivo do *perceptron* é encontrar uma função que mapeie corretamente as entradas para as saídas desejadas, minimizando o erro entre as previsões do modelo e os valores reais, isso é feito calculando o gradiente de uma função de custo e ajustando os pesos e os vieses da rede.

Quandos os pesos e viéses são atualizados é utilizado o termo época para se referir a uma iteração completa sobre o conjunto de dados de treinamento.

Para esclarecer o funcionamento do *perceptron* e do algoritmo de aprendizagem é proposto o Exemplo 1.1.

**Exemplo 1.1.** *Um investidor busca decidir qual ação adquirir com base nas variáveis: valor da ação, confiabilidade e volatilidade. Para isso, dispõe das avaliações de quatro colegas que também analisam essas mesmas características antes de realizar seus investimentos.*

Na Tabela 1.1 são apresentadas as avaliações dos colegas e a decisão final de compra, ou seja, o valor da ação é considerado baixo ou alto, a confiabilidade é considerada baixa ou alta, a volatilidade é considerada baixa ou alta e se comprou ou não, com 0 representando “não” e 1 representando “sim”,  $x^{(1)}, \dots, x^{(4)}$  representando as respostas de cada colega e  $y = (y_1, y_2, y_3, y_4)$  a decisão final de cada um.

Tabela 1.1: Banco de dados com as avaliações dos colegas e a decisão final de compra.

|           | valor da ação | confiabilidade | volatilidade | $y$ |
|-----------|---------------|----------------|--------------|-----|
| $x^{(1)}$ | 0             | 1              | 0            | 1   |
| $x^{(2)}$ | 1             | 0              | 1            | 0   |
| $x^{(3)}$ | 1             | 1              | 1            | 0   |
| $x^{(4)}$ | 0             | 0              | 0            | 1   |

Com isso, o *perceptron* tem três variáveis de entrada e como saída tem-se a decisão sobre a compra da ação. Os pesos iniciais são definidos aleatoriamente como:  $w = (w_1, w_2, w_3) = (0, 0, 0)$  e o viés como  $b = 0$ , utilizando a função de ativação *sigmoid* (1.3), com  $\alpha = 1$ . Para o caso geral, com  $x^{(i)} = (x_1^{(i)}, x_2^{(i)}, x_3^{(i)})$ , têm-se:

$$\hat{y}_i = \sigma(z_i), \text{ em que } z_i = \sum_{j=1}^3 x_j^{(i)} \cdot w_j + b$$

para  $i = 1, \dots, 4$ , em que  $\hat{y}_i$  é a resposta da rede, considerando  $\hat{y} = (\hat{y}_1, \hat{y}_2, \hat{y}_3, \hat{y}_4)$ . Testando para o primeiro dado de entrada, ou seja,  $x^{(1)} = (0, 1, 0)$  e  $y_1 = 1$ :

$$\begin{cases} z_1 = \sum_{j=1}^3 x_j^{(1)} \cdot w_j + b = 0 \cdot 0 + 1 \cdot 0 + 0 \cdot 0 + 0 = 0 \\ \hat{y}_1 = \sigma(0) = 0,5 \end{cases}.$$

A saída da função de ativação pode ser interpretada como a probabilidade de compra da ação, ou seja, a rede prevê que a probabilidade de compra da ação é de 50%. Para que essa probabilidade seja interpretada como uma decisão binária, define-se um limiar, do inglês *threshold*. Neste exemplo, adota-se *threshold* igual a 0,5 e a convenção de que a rede prevê compra somente quando  $\hat{y}_i > 0,5$ ; caso contrário, prevê não compra. Com isso, para o primeiro dado de entrada, a rede prevê  $\hat{y}_1 = 0$ .

Como  $y_1 = 1 \neq \hat{y}_1 = 0$ , o *perceptron* comete um erro e é necessário aplicar o algoritmo de aprendizagem, a unidade de McCulloch e Pitts encerraria na etapa anterior. Para isso, define-se uma função de custo, responsável por quantificar o erro cometido durante o treinamento. Neste exemplo, utiliza-se o Erro Quadrático Médio, dado por:

$$C(w, b) = \frac{\sum_{i=1}^4 (y_i - \hat{y}_i)^2}{n}, \quad (1.5)$$

com  $n$  representando a quantidade de dados presentes no conjunto de treinamento. Desta forma,  $C(w, b)$  não depende de  $w$  e  $b$  explicitamente, mas sim por meio de  $\hat{y}_i$ , ou seja:

$$\begin{cases} C(w, b) &= C(\hat{y}_i(w, b)) \\ \hat{y}_i &= \sigma(z_i) \\ z_i &= \sum_{j=1}^3 x_j^{(i)} \cdot w_j + b \end{cases}.$$

O processo de treinamento ajusta iterativamente os pesos e o viés por meio do método da GD, do inglês *Gradient Descent* (GD), cujo objetivo é minimizar a função de custo. Esse método utiliza as derivadas parciais da função de custo em relação aos pesos e ao viés, o que indica a direção de maior crescimento da função. Para reduzir o erro, atualizam-se os parâmetros na direção oposta ao gradiente. As derivadas são:

$$\frac{\partial C(w, b)}{\partial w_j} = -2 \cdot \frac{\sum_{i=1}^4 (y_i - \hat{y}_i) \cdot \sigma'(z_i) \cdot x_j^{(i)}}{n}; \quad (1.6)$$

$$\frac{\partial C(w, b)}{\partial b} = -2 \cdot \frac{\sum_{i=1}^4 (y_i - \hat{y}_i) \cdot \sigma'(z_i)}{n}. \quad (1.7)$$

No Exemplo 1.1 assume-se  $\sigma'(z_i) = 1$  para simplificação. Como a derivada aponta para a região de maior valor da função, multiplica-se por  $-1$  para obter a direção da região de minimização. O método de GD é apresentado no Algoritmo 1. A taxa de aprendizagem ( $\eta$ ) presente no Algoritmo 1 pode ser interpretada como o tamanho do passo dado em direção ao mínimo da função de custo, como mostrado na Figura 1.4. Quando o valor dessa taxa é muito pequeno, o processo de convergência torna-se mais lento, exigindo um número maior de iterações para atingir o mínimo. Por outro lado, se o passo for muito grande, o algoritmo pode ultrapassar o ponto de mínimo, dificultando ou até impedindo a convergência adequada, para o Exemplo 1.1 é utilizada uma taxa de aprendizado de 0,1. Em cada etapa que o passo é dado, os pesos e o viés são atualizados, e a função de custo é recalculada. O processo continua até que a função de custo atinja um valor mínimo ou até que seja atingido um número máximo de iterações, conhecido como épocas.

**Entrada:** Conjunto de treinamento  $\{(x^{(q)}, y_q)\}_{q=1}^n$ , taxa de aprendizado  $\eta$ , pesos iniciais  $w_1, w_2, \dots, w_k$  e viés  $b$

**Saída:** Pesos e viés ajustados

**Repita**

**Para**  $q \leftarrow 1$  **Até**  $n$  **Faça**

$$z_q \leftarrow \sum_{m=1}^k w_m \cdot x_m^{(q)} + b;$$

$$\hat{y}_q \leftarrow \sigma(z_q);$$

**Fim**

$$C(w, b) \leftarrow \frac{1}{n} \sum_{q=1}^n (y_q - \hat{y}_q)^2;$$

**Para**  $m \leftarrow 1$  **Até**  $k$  **Faça**

$$\nabla w_m \leftarrow \frac{2}{n} \sum_{q=1}^n (y_q - \hat{y}_q) \cdot \sigma'(z_q) \cdot x_m^{(q)};$$

**Fim**

$$\nabla b \leftarrow \frac{2}{n} \sum_{q=1}^n (y_q - \hat{y}_q) \cdot \sigma'(z_q);$$

**Para**  $m \leftarrow 1$  **Até**  $k$  **Faça**

$$w_m \leftarrow w_m + \eta \nabla w_m;$$

**Fim**

$$b \leftarrow b + \eta \nabla b;$$

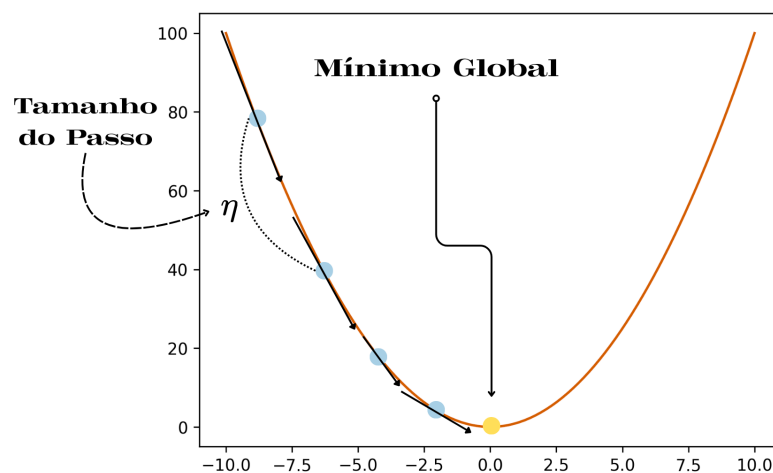
**até** critério de parada (convergência ou número máximo de épocas);

**Retorne**  $w_1, w_2, \dots, w_k, b$ ;

**Algoritmo 1:** Método da GD.

Na Figura 1.4 é apresentada uma ilustração da taxa de aprendizagem, em que o ponto amarelo representa o ponto de mínimo da função de custo, representada pela cor laranja, os pontos azuis representam o momento em que é feito o ajuste dos pesos e do viés e as setas pretas indicam a direção inversa do gradiente.

Figura 1.4: Representação da taxa de aprendizagem.



Fonte: Elaborado pelo autor.

Aplicando o Algoritmo 1 no Exemplo 1.1, com taxa de aprendizado  $\eta = 0,1$  e assumindo  $\sigma'(z_i) = 1$ , atualizando os pesos e o viés três vezes, é obtido o seguinte resultado.

Na época 1 de atualização, como  $w^{(0)} = (0, 0, 0)$  e  $b^{(0)} = 0$ , tem-se  $z_i = 0$  para todo  $i$ , logo

$$\hat{y} = (0, 5, 0, 5, 0, 5, 0, 5),$$

em que  $w^{(0)}$  e  $b^{(0)}$  são os pesos e viés iniciais. Calculando os gradientes,  $\nabla w_i$ , para  $i = 1, \dots, 3$  e  $\nabla b$ , têm-se:

$$\nabla w_1 = 2 \cdot \left( \frac{0,5 \cdot 0 + (-0,5) \cdot 1 + (-0,5) \cdot 1 + 0,5 \cdot 0}{4} \right) = -0,5,$$

$$\nabla w_2 = 2 \cdot \left( \frac{0,5 \cdot 1 + (-0,5) \cdot 0 + (-0,5) \cdot 1 + 0,5 \cdot 0}{4} \right) = 0,$$

$$\nabla w_3 = 2 \cdot \left( \frac{0,5 \cdot 0 + (-0,5) \cdot 1 + (-0,5) \cdot 1 + 0,5 \cdot 0}{4} \right) = -0,5,$$

$$\nabla b = 2 \cdot \left( \frac{0,5 - 0,5 - 0,5 + 0,5}{4} \right) = 0.$$

Portanto, os novos pesos são:  $w^{(1)} = w^{(0)} + 0,1 \cdot (\nabla w_1, \nabla w_2, \nabla w_3) = (-0,05, 0, -0,05)$  e o novo viés é:  $b^{(1)} = b^{(0)} + 0,1 \cdot \nabla b = 0$ .

Na época 2, com  $w^{(1)} = (-0,05, 0, -0,05)$  e  $b^{(1)} = 0$ , obtém-se,

$$z = (0, -0,1, -0,1, 0),$$

consequentemente,  $\hat{y} \approx (0,5, 0,4750, 0,4750, 0,5)$ . Com isso, os gradientes são:

$$(\nabla w_1, \nabla w_2, \nabla w_3, \nabla b) \approx (-0,2438, 0,0031, -0,2438, 0,0062).$$

Após a atualização,  $w^{(2)} \approx (-0,0494, 0,0003, -0,0494)$  e  $b^{(2)} \approx 0,0006$ .

Na época 3, usando  $w^{(2)}$  e  $b^{(2)}$ , tem-se,

$$z \approx (0,0009, -0,0981, -0,0978, 0,0006).$$

Então,  $\hat{y} \approx (0,5002, 0,4755, 0,4756, 0,5002)$ . Assim, os gradientes são,

$$(\nabla w_1, \nabla w_2, \nabla w_3, \nabla b) \approx (-0,2378, 0,0060, -0,2378, 0,0121),$$

de modo que  $w^{(3)} \approx (-0,0732, 0,0009, -0,0732)$  e  $b^{(3)} \approx 0,0018$ .

Verificando as previsões da rede após a terceira época, obtém-se:

$$x^{(1)} = (0, 1, 0) \Rightarrow z_1 \approx 0,0028 \Rightarrow \sigma(z_1) \approx 0,5007 \Rightarrow \hat{y}_1 = 1;$$

$$x^{(2)} = (1, 0, 1) \Rightarrow z_2 \approx -0,1445 \Rightarrow \sigma(z_2) \approx 0,4639 \Rightarrow \hat{y}_2 = 0;$$

$$x^{(3)} = (1, 1, 1) \Rightarrow z_3 \approx -0,1435 \Rightarrow \sigma(z_3) \approx 0,4642 \Rightarrow \hat{y}_3 = 0;$$

$$x^{(4)} = (0, 0, 0) \Rightarrow z_4 \approx 0,0018 \Rightarrow \sigma(z_4) \approx 0,5005 \Rightarrow \hat{y}_4 = 1.$$

Assim, após três épocas de atualização, os parâmetros ajustados classificam corretamente todos os exemplos do conjunto de treinamento. Para um dado de entrada fora do banco de dados,  $x = (1, 0, 0)$ , ou seja, o valor da ação é alto, a confiabilidade é baixa e a volatilidade é baixa, a rede prevê

$$\begin{cases} z \approx -0,0732 \cdot 1 + 0,0009 \cdot 0 - 0,0732 \cdot 0 + 0,0018 = -0,0713 \\ \sigma(z) \approx 0,4822 \Rightarrow \hat{y} = 0 \end{cases},$$

ou seja, a rede prevê que a ação não deve ser comprada.

Comumente o banco de dados é dividido em dois conjuntos: o conjunto de treinamento e o conjunto de teste. O conjunto de treinamento é utilizado para ajustar os pesos e o viés do *perceptron*, enquanto o conjunto de teste é aplicado para avaliar o desempenho do modelo em dados não empregados no treinamento. Essa divisão é essencial para garantir que o modelo aprenda a generalizar bem para novos dados, evitando o problema do *overfitting*, visto na Seção 1.2.3, que ocorre quando o modelo se ajusta excessivamente aos dados de treinamento, perdendo a capacidade de generalização.

## 1.2 Redes Neurais

Nesta seção são apresentados os conceitos relacionados às redes neurais, os quais são fundamentais para o desenvolvimento deste trabalho. Os conceitos apresentados nesta seção são utilizados na construção da rede neural proposta na Seção 3.1.

As redes neurais consistem em um “encadeamento” de múltiplas unidades, em que a saída de uma unidade é a entrada de outra. Redes neurais podem ser organizadas em camadas: uma *camada de entrada*, uma ou mais *camadas ocultas* e uma *camada de saída*. Cada camada é composta por várias unidades e cada unidade em uma camada está conectada a todas as unidades na camada seguinte. Na Figura 1.5(a) é ilustrada a estrutura básica de uma unidade vista na Seção 1.1.2. Na Figura 1.5(b) é mostrada uma rede neural com duas camadas ocultas, em que a camada de entrada tem três unidades, a primeira camada oculta tem cinco unidades, a segunda camada oculta tem quatro unidades e a camada de saída tem duas unidades. Matematicamente, é possível representar uma rede neural com uma camada de entrada com 3 unidades, duas camadas ocultas, cada camada contendo 5 e 4 unidades, respectivamente e uma saída da seguinte maneira:

$$\begin{aligned} &\text{definindo } \hat{y}_1^{(4)} = \sigma_1^{(4)} \left( x_1^{(4)} \right); \\ &\text{substituindo a equação (1.9) na equação (1.8);} \end{aligned} \tag{1.8}$$

$$x_1^{(4)} = \sum_{i^{(3)=1}}^4 w_1^{(3)}(i^{(3)}) \cdot x_{i^{(3)}}^{(3)} + b_1^{(4)}, \quad (1.9)$$

substituindo a equação (1.10) na equação (1.9);

$$x_{i^{(3)}}^{(3)} = \sigma_{i^{(3)}}^{(3)} \left( \sum_{i^{(2)=1}}^5 w_{i^{(3)}}^{(2)}(i^{(2)}) \cdot x_{i^{(2)}}^{(2)} + b_{i^{(3)}}^{(3)} \right), \quad (1.10)$$

substituindo a equação (1.11) na equação (1.10);

$$x_{i^{(2)}}^{(2)} = \sigma_{i^{(2)}}^{(2)} \left( \sum_{i^{(1)=1}}^3 w_{i^{(2)}}^{(1)}(i^{(1)}) \cdot x_{i^{(1)}}^{(1)} + b_{i^{(2)}}^{(2)} \right), \quad (1.11)$$

em que  $x_{i^{(1)}}^{(1)}$  representa a  $i^{(1)}$ -ésima unidade da camada de entrada,  $x_{i^{(2)}}^{(2)}$  representa a  $i^{(2)}$ -ésima unidade da segunda camada ou da primeira camada oculta,  $x_{i^{(3)}}^{(3)}$  representa a  $i^{(3)}$ -ésima unidade da segunda camada oculta e  $\hat{y}_1^{(4)}$  representa a saída da rede. Os pesos e os vieses são representados por  $w_{i^{(l)}}^{(l-1)}(i^{(l-1)})$  e  $b_{i^{(l)}}^{(l)}$ , respectivamente, em que  $l$  representa o número da camada, para  $l = 2, 3, 4$ . As funções de ativação são representadas por  $\sigma_{i^{(l)}}^{(l)}$ , para  $l = 2, 3, 4$ .

De forma geral, para uma rede neural com  $L$  camadas, a saída da rede pode ser expressa pela seguinte equação:

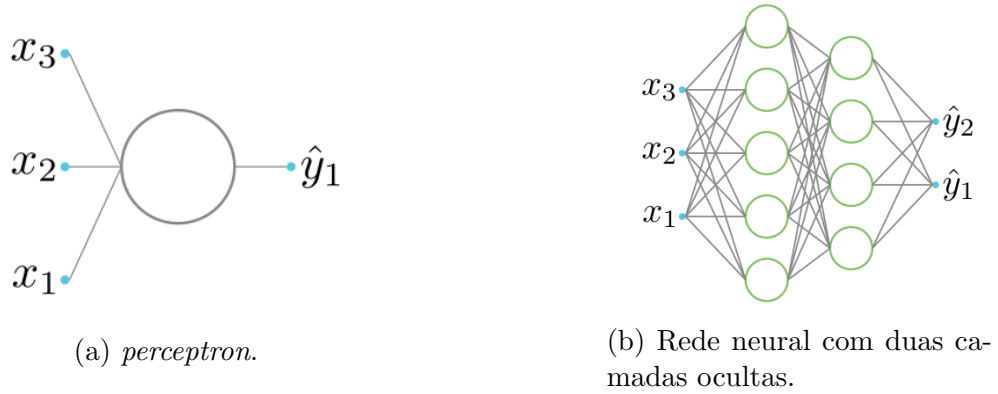
$$\begin{aligned} \hat{y}_1^{(L)} &= \sigma_1^{(L)} \left( x_1^{(L)} \right); \\ x_1^{(L)} &= \sum_{i^{(L-1)=1}}^{n_{L-1}} w_1^{(L-1)}(i^{(L-1)}) \cdot x_{i^{(L-1)}}^{(L-1)} + b_1^{(L)}; \\ x_{i^{(L-1)}}^{(L-1)} &= \sigma_{i^{(L-1)}}^{(L-1)} \left( \sum_{i^{(L-2)=1}}^{n_{L-2}} w_{i^{(L-1)}}^{(L-2)}(i^{(L-2)}) \cdot x_{i^{(L-2)}}^{(L-2)} + b_{i^{(L-1)}}^{(L-1)} \right); \\ &\vdots \\ x_{i^{(2)}}^{(2)} &= \sigma_{i^{(2)}}^{(2)} \left( \sum_{i^{(1)=1}}^{n_1} w_{i^{(2)}}^{(1)}(i^{(1)}) \cdot x_{i^{(1)}}^{(1)} + b_{i^{(2)}}^{(2)} \right); \end{aligned} \quad (1.12)$$

em que  $n_l$  representa a quantidade de unidades na  $l$ -ésima camada, para  $l = 1, 2, \dots, L-1$ . Na Figura 1.5 é possível visualizar a estrutura visual básica de uma unidade gerando uma saída e uma rede neural com duas camadas ocultas, ilustrando a organização das unidades em camadas e as conexões entre elas.

Na Figura 1.5 os pesos e vieses foram ocultados para facilitar a visualização. Outra forma de representar uma rede neural é por meio da notação matricial, em que as entradas, os pesos e os vieses são representados por matrizes e vetores. Uma mudança de notação é feita para simplificar a representação,  $w_{i^{(l+1)}}^{(l)}(i^{(l)}) = w_{i^{(l)}i^{(l+1)}}^{(l)}$ . Com isso, para uma rede neural com  $L$  camadas, a saída da rede pode ser expressa pela seguinte equação matricial:

$$\begin{aligned} \hat{Y}_{L \times 1}^{(L)} &= \sigma_{L \times 1}^{(L)} \left( X_{L \times 1}^{(L)} \right); \\ X_{L \times 1}^{(L)} &= W_{(L \times L-1)}^{(L-1)} \odot X_{(L-1 \times 1)}^{(L-1)} + B_{(L \times 1)}^{(L)}; \end{aligned} \quad (1.13)$$

Figura 1.5: Estrutura visual básica de uma unidade e uma rede neural com duas camadas ocultas.



Fonte: Elaborado pelo autor.

em que  $W^{(L-1)}$  é a matriz de pesos da  $(L-1)$ -ésima camada,  $X^{(L)}$  é o vetor de entradas da  $(L)$ -ésima camada,  $B^{(L)}$  é o vetor de vieses da  $L$ -ésima camada,  $\hat{Y}_{L \times 1}^{(L)}$  é o vetor de saídas e  $\odot$  representa a multiplicação matricial, ou seja:

$$x_1^{(L)} = \underbrace{\begin{bmatrix} w_{11}^{(L-1)} & w_{12}^{(L-1)} & \cdots & w_{1(L-1)}^{(L-1)} \\ w_{21}^{(L-1)} & w_{22}^{(L-1)} & \cdots & w_{2(L-1)}^{(L-1)} \\ \vdots & \vdots & \ddots & \vdots \\ w_{L1}^{(L-1)} & w_{L2}^{(L-1)} & \cdots & w_{L(L-1)}^{(L-1)} \end{bmatrix}}_{L \times (L-1)} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{L-1} \end{bmatrix}}_{(L-1) \times 1} + \underbrace{\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_L \end{bmatrix}}_{L \times 1}.$$

Um processo iterativo é feito para calcular  $X_{(L-1 \times 1)}^{(L-1)}$  semelhante ao processo apresentado na equação (1.12). Essa forma representa a saída da rede caso tenha apenas uma saída.

### 1.2.1 *Backpropagation*

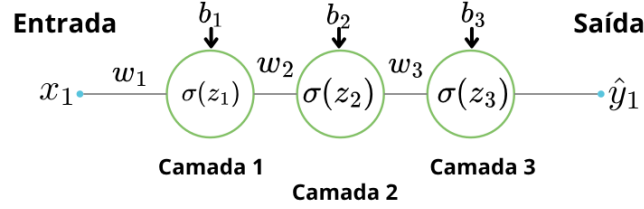
Assim como no *perceptron*, as redes neurais utilizam algoritmos de aprendizagem para ajustar seus pesos e vieses com o objetivo de minimizar uma função de custo. O algoritmo mais comum utilizado para esse propósito é o *backpropagation* (retropropagação) (HECHT-NIELSEN, 1992) e (WERBOS, 1990), que é uma extensão do método da GD utilizado no *perceptron*. O *backpropagation* é utilizado para calcular os gradientes da função de custo em relação aos pesos e vieses de todas as camadas da rede neural, permitindo a atualização eficiente desses parâmetros durante o treinamento. O algoritmo funciona propagando o erro da saída da rede de volta para as camadas anteriores, ajustando os pesos com base na contribuição de cada unidade para o erro total.

Esse processo é repetido iterativamente até que a função de custo seja minimizada, resultando em uma rede neural treinada. Essa propagação do erro é feita utilizando a regra da cadeia do cálculo diferencial, que permite calcular as derivadas parciais de funções compostas.

Na Figura 1.6 é ilustrada uma rede neural simples, em que  $x_1$  é a entrada da rede,  $w_i$ ,  $b_i$  e

$\sigma(z_i)$  é o peso, viés e a função de ativação da  $i$ -ésima camada, para  $i = 1, 2, 3$ , e  $\hat{y}_1$  é a saída da rede para  $x_1$ .

Figura 1.6: Ilustração do processo de *backpropagation* em uma rede neural.



Fonte: Elaborado pelo autor.

A equação (1.14) mostra o cálculo do gradiente para ajustar o peso da primeira camada ( $w_1$ ) utilizando a regra da cadeia:

$$\nabla_{w_1} C = \frac{\partial C(w, b)}{\partial w_1} = \frac{C(w, b)}{\sigma(z_3)} \cdot \frac{\sigma(z_3)}{z_3} \cdot \frac{z_3}{\sigma(z_2)} \cdot \frac{\sigma(z_2)}{z_2} \cdot \frac{z_2}{\sigma(z_1)} \cdot \frac{\sigma(z_1)}{z_1} \cdot \frac{z_1}{w_1}, \quad (1.14)$$

em que  $z_1 = w_1 \cdot x + b_1$ ,  $z_2 = \sigma(z_1) \cdot w_2 + b_2$  e  $z_3 = \sigma(z_2) \cdot w_3 + b_3$  são as saídas das camadas ocultas e  $C(w_i, b_i)$  é a função de custo do erro quadrático médio, para  $i = 1, 2, 3$ , ou seja,  $C(w, b) = \sum_{i=1}^n \frac{(y_i - \hat{y}_i)^2}{n}$ , em que  $y_i$  é a saída desejada,  $\hat{y}_i$  é a saída da rede e  $n$  é o número de amostras no conjunto de dados. O cálculo do gradiente para os outros pesos e vieses segue o mesmo processo, utilizando a regra da cadeia para propagar o erro de volta através da rede.

O algoritmo de *backpropagation* é dividido em dois passos, o passo para frente, do inglês *forward pass*, e o passo para trás, do inglês *backward pass*. O algoritmo pode ser resumido nos seguintes passos, apresentados no Algoritmo 2.

**Entrada:** Dados de treinamento  $(x, y)$ , taxa de aprendizado  $\eta$ , Matriz de pesos  $W$  e vetor de vieses  $b$ ;

**Saída:** Matriz de pesos  $W$  e vetor de vieses  $b$  atualizados;

**Repita**

**Passo para frente:** propagar  $x$  pela rede (camada a camada) e obter  $\hat{y}$ ;

**Cálculo do Erro:** calcular o erro comparando  $\hat{y}$  com  $y$  usando uma função de custo;

**Passo para trás:** retropropagar o erro e calcular os gradientes  $\nabla_W C$  e  $\nabla_b C$  (regra da cadeia);

**até critério de parada** (*ex.:  $n^o$  de épocas ou convergência*);

**Algoritmo 2:** Etapas do treinamento via *backpropagation*.

O algoritmo de *backpropagation* não calcula os valores dos pesos e vieses diretamente, mas sim os gradientes da função de custo em relação a esses parâmetros. Esses gradientes são então utilizados por um método de otimização, como o método de GD, para atualizar os pesos e vieses da rede neural, com o objetivo de minimizar a função de custo e melhorar o desempenho do modelo.

## 1.2.2 Gradiente Descendente Estocástico

O Gradiente Descendente Estocástico, em inglês *Stochastic Gradient Descent* (SGD) é uma variação do método de GD tradicional, amplamente utilizado no treinamento de redes neurais e outros modelos de aprendizagem de máquina (RUDER, 2016). A principal diferença entre o SGD e o método de GD é a forma como os gradientes são calculados e atualizados durante o treinamento. No método de GD tradicional, os gradientes são calculados usando todo o conjunto de dados de treinamento. Assim, os pesos são atualizados uma vez em cada época de treinamento, o que pode ser computacionalmente caro e demorado, especialmente para grandes conjuntos de dados.

Em contraste, o SGD calcula os gradientes usando apenas uma amostra aleatória, ou um pequeno lote, do inglês *mini-batch*, do conjunto de dados em cada iteração. Caso o conjunto de dados tenha um tamanho de 1000 e o tamanho do lote seja 10, o SGD atualiza os pesos 100 vezes por época. Este processo de atualização dos pesos é mais rápido e eficiente. Apesar da GD padrão fornecer uma direção mais precisa para o mínimo global da função de custo, o SGD é capaz de chegar próximo do mínimo de forma mais rápida e eficiente.

Na Figura 1.7(a) é apresentado o funcionamento do SGD, em que cada seta preta indica uma atualização dos pesos do modelo. Na Figura 1.7(b) é apresentado o caminho percorrido pelo método de GD tradicional.

Figura 1.7: Comparação entre os métodos de GD.



Fonte: Elaborado pelo autor.

O gráfico da Figura 1.7 representa a função de custo em um espaço de parâmetros.

### 1.2.3 *Overfitting*

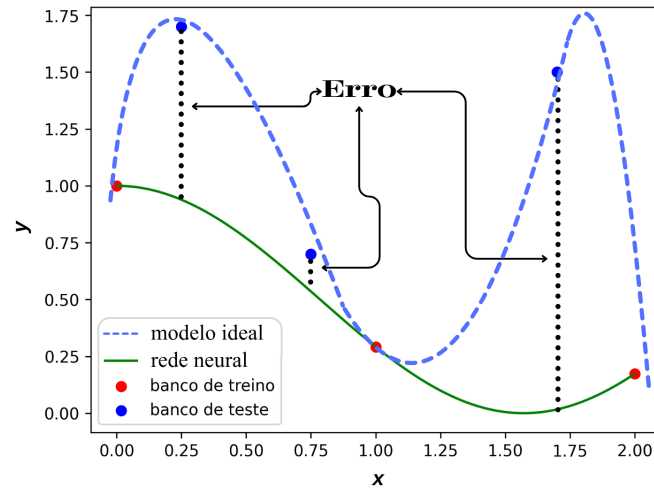
O *overfitting*, em português “Sobreajuste” ou “Ajuste Excessivo”, é um problema comum em aprendizagem de máquina, em que um modelo aprende os detalhes e o ruído dos dados de treinamento a tal ponto que este se torna incapaz de generalizar para novos dados (DEEP LEARNING BOOK, 2025b). Quando ocorre *overfitting*, o modelo tem um desempenho excelente nos dados de treinamento, mas falha ao fazer previsões precisas em dados não utilizados anteriormente.

O *overfitting* ocorre quando o modelo é muito complexo em relação à quantidade e qualidade dos dados disponíveis para treinamento. Modelos com muitos parâmetros, como redes neurais profundas, são particularmente suscetíveis ao *overfitting*, especialmente quando o conjunto de dados de treinamento é pequeno ou contém ruído significativo. Na Figura 1.8 é apresentada uma ilustração do *overfitting*, em que os pontos vermelhos são os dados de treinamento e os pontos azuis são os dados de teste. A curva verde representa um modelo que se ajusta perfeitamente aos dados de treinamento, mas falha em generalizar para novos dados, enquanto a curva azul seria um modelo ideal mais geral.

### 1.2.4 Hiperparâmetros

Um algoritmo de aprendizado pode ser visto como uma função que recebe dados de treinamento como entrada e produz como saída um modelo matemático. No entanto, na prática muitos algoritmos de aprendizado possuem parâmetros adicionais que não são aprendidos diretamente a partir dos dados de treinamento, mas que influenciam o processo de aprendizado. Esses parâmetros são chamados de hiperparâmetros e sua escolha adequada é crucial para o desempenho do modelo (BENGIO, 2012). Alguns exemplos comuns de hiperparâmetros incluem:

- Taxa de Aprendizado: Controla o tamanho dos passos dados durante a atualização dos

Figura 1.8: Ilustração do *overfitting*.

Fonte: Elaborado pelo autor.

pesos do modelo. Uma taxa de aprendizado muito alta pode levar a oscilações e impedir a convergência, enquanto uma taxa muito baixa pode resultar em um processo de treinamento lento. Valores típicos para a taxa de aprendizado variam entre 0,01 e 0,1, dependendo do problema e do algoritmo utilizado (BENGIO, 2012).

- **Número de Épocas:** Define quantas vezes o algoritmo de aprendizado passará por todo o conjunto de dados de treinamento. Um número muito alto pode levar ao *overfitting*, enquanto um número muito baixo pode resultar em *underfitting* (subajuste). Uma técnica comum para determinar o número ideal de épocas é o uso do *Early Stopping* (do inglês Parada Antecipada), que monitora o desempenho do modelo em um conjunto de validação e interrompe o treinamento quando o desempenho começa a piorar (BENGIO, 2012).
- **Tamanho do Lote:** Refere-se ao número de amostras de dados usadas para calcular o gradiente em cada iteração do treinamento. Tamanhos de lote menores podem levar a atualizações mais ruidosas, enquanto tamanhos maiores podem resultar em um treinamento mais estável, embora mais lento. Os valores escolhidos geralmente variam entre 1 e 32 (BENGIO, 2012).
- **Arquitetura do Modelo:** Inclui o número de camadas, o número de unidades em cada camada e o tipo de funções de ativação utilizadas. Modelos mais complexos podem capturar padrões mais sofisticados, mas também são mais propensos ao *overfitting* (BENGIO, 2012).

### 1.2.5 Regularização

A regularização é uma técnica utilizada para prevenir o *overfitting* em modelos de aprendizagem de máquina (DEEP LEARNING BOOK, 2025c). Existem várias técnicas de regularização, sendo as mais comuns:

- Regularização L1 (Norma 1): Adiciona uma penalidade proporcional à soma dos valores absolutos dos pesos à função de custo. Esta técnica pode levar a soluções esparsas, onde alguns pesos são reduzidos a zero. A equação da função de custo com regularização L1 é dada por:

$$C(w, b) = C_0(w, b) + \frac{\lambda}{n} \sum_i |w_i|, \quad (1.15)$$

em que  $C_0(w, b)$  é a função de custo original,  $\lambda$  é o parâmetro de regularização que controla a força da penalidade,  $w_i$  são os pesos do modelo,  $b$  é o viés e  $n$  é o número de amostras no conjunto de dados (DEEP LEARNING BOOK, 2025c).

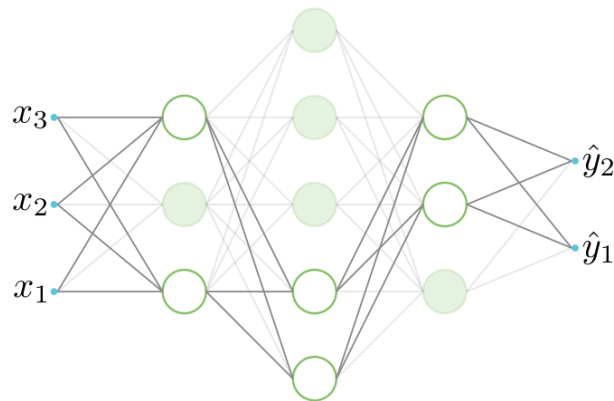
- Regularização L2 (Norma 2): Adiciona uma penalidade proporcional à soma dos quadrados dos pesos à função de custo. Este procedimento incentiva a distribuir os pesos de forma mais uniforme no modelo, evitando que alguns pesos se tornem muito grandes. A equação da função de custo com regularização L2 é dada por:

$$C(w, b) = C_0(w, b) + \frac{\lambda}{n} \sum_i w_i^2, \quad (1.16)$$

em que os termos são definidos da mesma forma que na regularização L1 (DEEP LEARNING BOOK, 2025c).

- *Dropout*: Durante o treinamento, essa técnica desativa aleatoriamente uma fração das unidades em cada camada da rede neural para cada época. Esta técnica força a rede a aprender representações mais robustas e reduz a dependência de unidades específicas, ajudando a prevenir o *overfitting*. Em termos gerais, ao aplicar o *dropout*, em português, descarte, estão sendo treinadas uma coleção de sub-redes dentro da rede original, o que pode promover a generalização do modelo (DEEP LEARNING BOOK, 2025c). Na Figura 1.9 é ilustrado o funcionamento do *dropout*, em que 45% do total das unidades são removidas aleatoriamente durante uma época de treinamento, as unidades removidas são representadas por círculos com interior verde, enquanto as unidades ativas são representadas por círculos com interior branco.

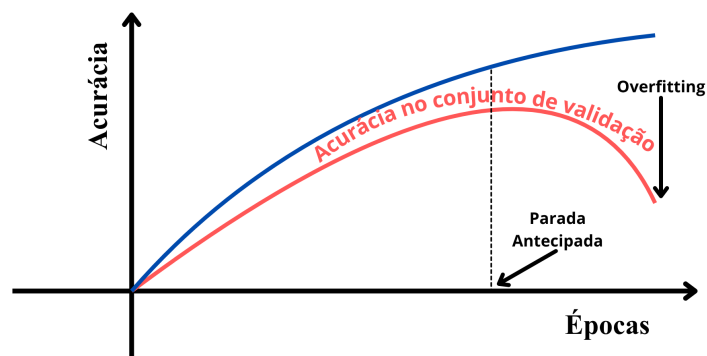
Figura 1.9: Ilustração do *Dropout* removendo 45% das unidades aleatoriamente durante uma época de treinamento.



Fonte: Elaborado pelo autor.

- *Early Stopping* (Parada Antecipada): Monitora o desempenho do modelo em um conjunto de validação durante o treinamento e interrompe o processo quando o desempenho começa a piorar. Este procedimento evita que o modelo continue a se ajustar aos dados de treinamento além do ponto ideal. Como ilustrado na Figura 1.10, o ponto de parada ideal é onde a acurácia na validação começa a diminuir, indicando o início do *overfitting*.

Figura 1.10: Ilustração do *Early Stopping*.



Fonte: Adaptado de DEEP LEARNING BOOK (2025c).

### 1.2.6 *Momentum* e Adam

O *Momentum*, do português Momento, é uma técnica utilizada para acelerar o processo de treinamento de redes neurais, especialmente em situações onde a função de custo apresenta regiões íngremes ou planas (SUTSKEVER, 2013). A ideia principal do *momentum* é acumular uma média ponderada dos gradientes anteriores para suavizar as atualizações dos pesos, permitindo que o modelo mantenha uma direção consistente durante o treinamento. Este procedimento

ajuda a evitar oscilações e acelera a convergência para o mínimo da função de custo. Dada uma função de custo  $C(w, b)$  a ser minimizada, o *momentum* clássico é definido por:

$$\begin{cases} v_{t+1} = \beta v_t + \epsilon \nabla C(w_i, b_i) \\ \theta_{t+1} = \theta_t + v_{t+1} \end{cases}, \quad (1.17)$$

em que  $\epsilon > 0$  é a taxa de aprendizagem,  $\beta \in [0, 1]$  é o coeficiente de *momentum*,  $v_t$  é a velocidade acumulada no tempo  $t$ ,  $\theta_t = (w_i, b_i)$  representa os parâmetros do modelo (pesos e vieses) no tempo  $t$  e  $\nabla C(w_i, b_i)$  é o gradiente da função de custo em relação aos parâmetros do modelo. O coeficiente de *momentum*  $\beta$  controla a influência dos gradientes anteriores nas atualizações atuais. *Adaptive Moment Estimation* (Adam) é um método que calcula taxas de aprendizagem ( $\eta$ ) adaptativas para cada parâmetro do modelo, em cada passo de atualização dos pesos durante o treinamento. O Adam combina as vantagens do *momentum* e do método de GD (RUDER, 2016). Atualmente, o Adam é um dos algoritmos de otimização mais populares para treinamento de redes neurais, devido à sua eficiência e capacidade de lidar com grandes conjuntos de dados e parâmetros.

### 1.2.7 *Vanishing Gradient*

O problema do *vanishing gradient*, do português gradiente que desaparece, ocorre durante o treinamento de redes neurais profundas, em que os gradientes calculados durante o processo de *backpropagation* tornam-se extremamente pequenos à medida que são propagados para as camadas mais próximas da entrada. Este problema acontece devido à multiplicação repetida de pequenas derivadas associadas às funções de ativação e aos pesos das camadas. Como resultado, os pesos das camadas iniciais da rede recebem atualizações insignificantes, dificultando o aprendizado efetivo dessas camadas. Como neste trabalho serão utilizadas redes neurais profundas, é importante entender o impacto do problema do *vanishing gradient* e as técnicas para mitigá-lo.

**Exemplo 1.2.** *Considere uma rede neural profunda com  $N$  camadas ocultas, em que todas as unidades em cada camada utilizam a função de ativação sigmoid logística, definida como:*

$$\sigma(z) = \frac{1}{1 + e^{-z}}. \quad (1.18)$$

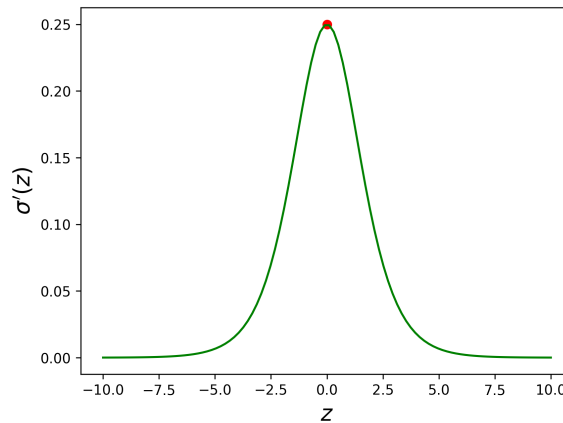
A derivada dessa função é dada por:

$$\sigma'(z) = \sigma(z) \cdot (1 - \sigma(z)). \quad (1.19)$$

Durante o processo de *backpropagation*, o gradiente da função de custo em relação aos pesos das camadas iniciais é calculado utilizando a regra da cadeia. Este processo envolve a multiplicação das derivadas das funções de ativação de cada camada, como mostra a equação (1.14). Como a derivada da função sigmoid é sempre menor que 0,25, o valor máximo ocorre em  $z = 0$ , como mostrado na Figura 1.11. A multiplicação repetida dessas pequenas derivadas resulta em

gradientes que se tornam cada vez menores à medida que são propagados para trás através das camadas da rede. Por exemplo, se uma rede possui 10 camadas ocultas, o gradiente pode ser reduzido a um valor extremamente pequeno, como  $10^{-9}$  ou menor, tornando as atualizações dos pesos nas camadas iniciais praticamente insignificantes.

Figura 1.11: Gráfico da derivada da função sigmoid logística.



Fonte: Elaborado pelo autor.

O problema do *vanishing gradient* é particularmente comum em redes neurais que utilizam funções de ativação como a *sigmoid* ou a tangente hiperbólica. Para contornar o problema do *vanishing gradient*, várias técnicas podem ser empregadas, incluindo:

- **Funções de Ativação Alternativas:** Funções de ativação como a ReLU e suas variantes (Leaky ReLU, Parametric ReLU) têm derivadas constantes para valores positivos, o que ajuda a manter os gradientes maiores durante o *backpropagation*.
- **Batch Normalization:** Normaliza as ativações de cada camada durante o treinamento, o que pode ajudar a estabilizar os gradientes e acelerar o processo de aprendizagem.

### 1.2.8 Função de Custo *Binary Cross-Entropy*

O *Vanishing Gradient* acontece principalmente em redes neurais profundas que utilizam funções de ativação como a *sigmoid*, uma das formas de contornar esse problema é utilizando a função de custo *Binary Cross-Entropy* (BCE), que é comumente utilizada em tarefas de classificação binária (ELHARROUSS *et al.*, 2025). A função de custo BCE é definida como:

$$\text{BCE}(y, \hat{y}) = -\frac{1}{n} \sum_{i=1}^n [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)], \quad (1.20)$$

em que  $y_i$  é o valor de saída verdadeiro da  $i$ -ésima amostra (0 ou 1),  $\hat{y}_i$  é a previsão da rede para a  $i$ -ésima amostra (probabilidade de pertencer à classe 1) e  $n$  é o número total de amostras no conjunto de dados. A função de custo BCE é especialmente eficaz para lidar com o problema do *vanishing gradient*, pois penaliza fortemente as previsões incorretas, o que pode ajudar a manter os gradientes maiores durante o processo de *backpropagation*.

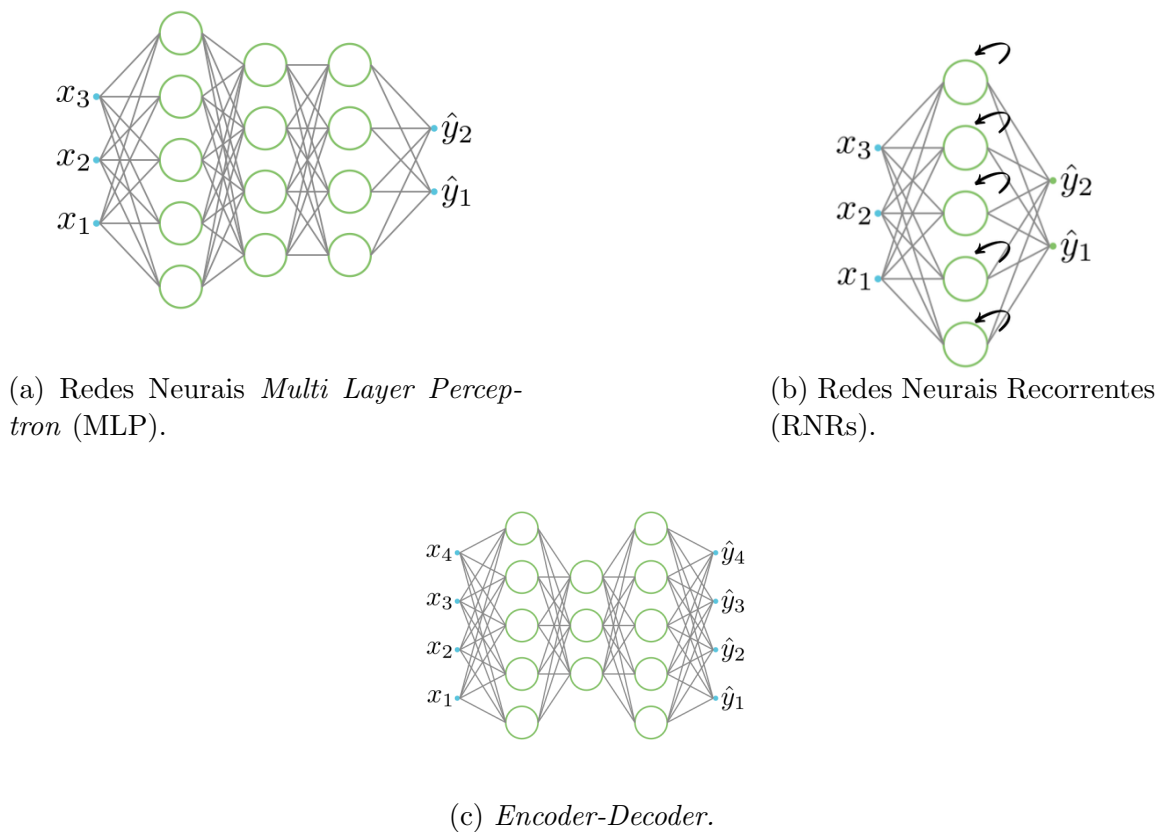
### 1.2.9 Arquiteturas de Redes Neurais

Existem diversas arquiteturas de redes neurais, cada uma projetada para lidar com diferentes tipos de problemas e dados. Algumas das arquiteturas mais comuns incluem (DEEP LEARNING BOOK, 2025a):

- Redes Neurais *perceptron* Multicamadas, do inglês, *Multi Layer Perceptron* (MLP): São redes neurais *feedforward*, ou seja, a entrada passa por uma ou mais camadas ocultas até ser processada pela camada de saída. Cada unidade em uma camada oculta está conectada a todas as unidades na camada seguinte, e as informações fluem apenas em uma direção, da entrada para a saída. Após cada saída a rede ajusta os pesos utilizando o algoritmo de *backpropagation*. MLPs são amplamente utilizadas para tarefas de classificação e regressão (DEEP LEARNING BOOK, 2025a).
- Redes Neurais Convolucionais (RNCs): São projetadas para processar dados com uma estrutura de grade, como imagens. Estas utilizam operações de convolução para extrair características espaciais dos dados de entrada. RNCs são amplamente utilizadas em tarefas de visão computacional, como reconhecimento de objetos e detecção de faces (DEEP LEARNING BOOK, 2025a).
- Redes Neurais Recorrentes (RNRs): São projetadas para lidar com dados sequenciais, onde a saída em um determinado instante depende das entradas anteriores. RNRs possuem conexões recorrentes que permitem que informações sejam mantidas ao longo do tempo. Estas são amplamente utilizadas em tarefas como processamento de linguagem natural e reconhecimento de fala (DEEP LEARNING BOOK, 2025a).
- Redes Neurais Generativas Adversariais (NGAs): Consistem em duas redes neurais que competem entre si: um gerador e um discriminador. O gerador cria dados sintéticos, enquanto o discriminador tenta distinguir entre dados reais e sintéticos. NGAs são amplamente utilizadas para geração de imagens realistas e outras tarefas criativas (DEEP LEARNING BOOK, 2025a).
- *Encoder-Decoder*: São arquiteturas que consistem em duas partes principais: o *encoder*, que comprime os dados de entrada em uma representação latente, e o *decoder*, que reconstrói os dados a partir dessa representação. Essas redes são amplamente utilizadas em tarefas como tradução automática e geração de texto (DEEP LEARNING BOOK, 2025a).

Também é possível combinar diferentes arquiteturas de redes neurais, como abordado na Seção 3.1. Na Figura 1.12 são ilustradas três arquiteturas de redes neurais.

Figura 1.12: Arquiteturas de redes neurais.



Fonte: Elaborado pelo autor.

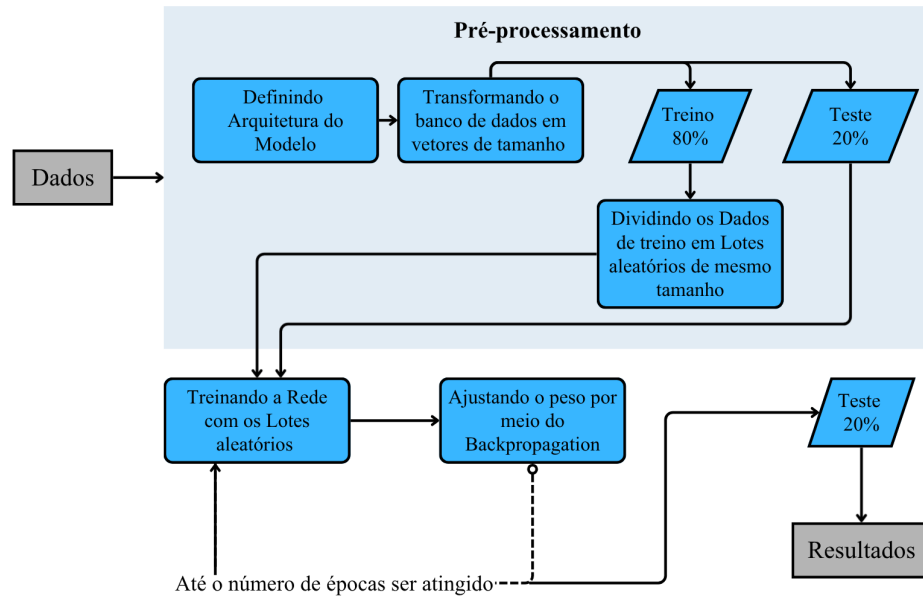
Nas Figuras 1.12(a) e 1.12(c) são ilustrados a arquitetura de uma rede neural MLP, com o funcionamento semelhante ao desenvolvido na equação (1.12). Na Figura 1.12(b) é mostrada a arquitetura de uma rede neural recorrente, estas redes possuem conexões recorrentes, ou seja, a saída de uma unidade é realimentada como entrada para a mesma unidade ou para unidades anteriores, permitindo que informações sejam mantidas ao longo do tempo, por não estar no escopo deste trabalho, não é abordado os detalhes matemáticos de funcionamento das RNNs.

### 1.2.10 Aplicação da Arquitetura MLP para Classificação de Dígitos Manuscritos

Para exemplificar o funcionamento de uma rede neural no reconhecimento de imagens, considere a tarefa de classificação de dígitos manuscritos utilizando o conjunto de dados MNIST (LECUN, 1998). Na Figura 1.13 é ilustrado o fluxo de dados durante o processo de treinamento e avaliação da rede neural para essa tarefa. O processo começa com a preparação dos dados, em que as imagens são pré-processadas, transformadas em vetores e normalizadas para garantir que os valores dos pixels estejam em uma faixa adequada para o treinamento da rede. Em seguida, os dados são divididos em conjuntos de treinamento e teste, onde o conjunto de treinamento é utilizado para ajustar os pesos da rede neural, enquanto o conjunto de teste é utilizado para

avaliar o desempenho do modelo após o treinamento. Em seguida é utilizado o algoritmo de SGD para atualizar os pesos da rede com base nos gradientes calculados pelo algoritmo de *backpropagation* e dividir o conjunto de dados de treinamento em lotes.

Figura 1.13: Fluxo de dados durante o processo de treinamento e avaliação da rede neural.



Fonte: Elaborado pelo autor.

O conjunto de dados MNIST consiste em 70.000 imagens em escala de cinza de dígitos manuscritos (0 a 9), cada uma com uma resolução de  $28 \times 28$  pixels, como mostrado na Figura 1.14.

Figura 1.14: Exemplos de dígitos manuscritos do conjunto de dados MNIST.

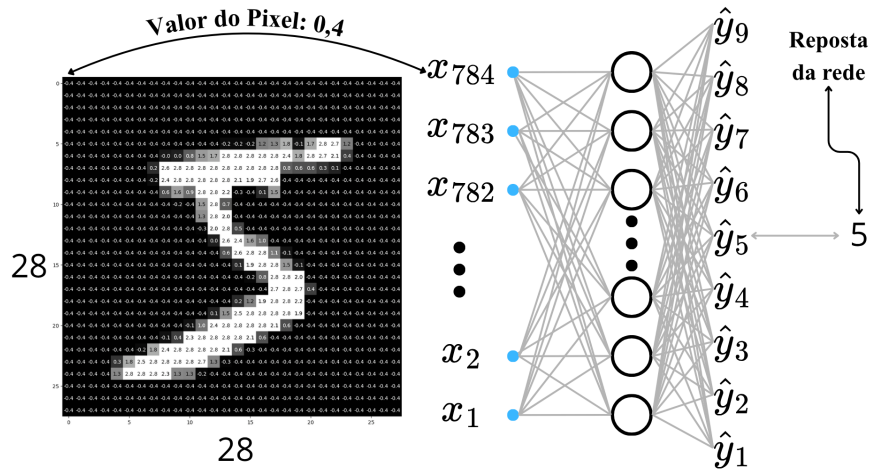


Fonte: Adaptado de LeCun (1998).

Cada imagem é representada como uma matriz de  $28 \times 28$ , em que cada pixel tem um valor de intensidade entre 0 (preto) e 255 (branco). Para o reconhecimento dos dígitos é utilizado uma rede MLP e os métodos presentes nas Seções 1.2.1, 1.2.2, 1.2.5 e 1.2.7 para o treinamento

e otimização dos pesos. Para compor a camada de entrada da rede neural, cada imagem é “achatada” em um vetor de 784 elementos ( $28 \times 28 = 784$ ), em que cada elemento corresponde à intensidade de um pixel. O objetivo da rede com camadas totalmente conectadas é aprender a classificar corretamente cada imagem em uma das 10 classes correspondentes aos dígitos de 0 a 9. Como a camada de saída da rede neural deve ter 10 unidades (uma para cada classe), a arquitetura da rede pode ser definida como  $[784, 30, 10]$ , onde a camada de entrada tem 784 unidades, a camada oculta tem 30 unidades, valor escolhido aleatoriamente, e a camada de saída tem 10 unidades, como mostrado na Figura 1.15.

Figura 1.15: Arquitetura da rede neural para classificação de dígitos manuscritos.



Fonte: Elaborado pelo autor.

Na Figura 1.15 uma imagem do dígito 5 é processada pela rede neural. A camada de entrada recebe o valor de cada entrada do vetor da imagem, com esses valores sendo processados pela camada oculta e a camada de saída fornecendo as probabilidades para cada classe de dígitos, sendo a classe com a maior probabilidade escolhida como a previsão da rede.

A função de ativação utilizada é a função *sigmoid* logística vista na equação (1.3), que é comumente usada em tarefas de classificação. A saída da rede é um vetor de 10 elementos, em que cada elemento representa a probabilidade da imagem pertencer a uma das classes de dígitos.

Para o treinamento da rede é utilizado o algoritmo de *backpropagation* e o SGD, com uma taxa de aprendizagem de 0,1 e a normalização *L1* visto na equação (1.15). O treinamento é realizado para 30 épocas. O banco de dados é dividido em um conjunto de treinamento (80%) com 60.000 imagens e um conjunto de teste (20%) com 10.000 imagens, divisão realizada aleatoriamente. Após o treinamento, a rede neural é avaliada no conjunto de teste para medir sua acurácia, que é a proporção de previsões corretas em relação ao total de previsões feitas:

$$\text{Acurácia} = \frac{\text{Número de Previsões Corretas}}{\text{Número Total de Previsões}}. \quad (1.21)$$

Com as configurações mencionadas, a rede neural alcançou uma acurácia de aproximada-

mente 95% no conjunto de teste, o que indica que esta foi capaz de aprender a classificar corretamente a maioria das imagens de dígitos manuscritos. Esta aplicação resultou em um trabalho apresentado no *I Workshop* de Matemática que foi realizado na Universidade Federal de Uberlândia (SENA; JAFELICE; SANTOS, 2025).

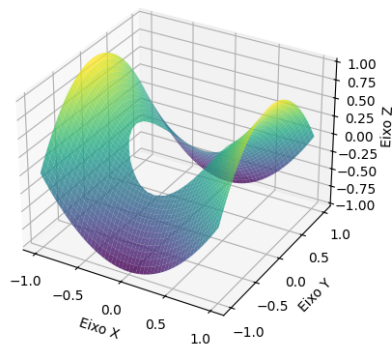
### 1.2.11 Aplicação da Arquitetura MLP para Reconstrução de Figura Geométrica

As redes neurais podem ser utilizadas para reconstruir malhas geométricas (SENA; JAFELICE; SANTOS, 2024). Neste aplicação, utilizou-se uma MLP com duas unidades na camada de entrada, dez unidades na camada oculta e uma unidade na camada de saída, para aproximar o gráfico do parabolóide hiperbólico  $z = x^2 - y^2$ . A rede é treinada utilizando o algoritmo de *backpropagation* e o SGD, com uma taxa de aprendizagem de 0,01 e a normalização *L2* visto na equação (1.16). O treinamento é realizado para 1000 épocas com todos as unidades tendo função de ativação *sigmoid* logística. Os dados considerados são nós de uma malha cartesiana uniforme  $x, y \in [-1, 1]$ , divididos em duzentas partes com  $z = x^2 - y^2$ . Neste modelo são considerados 20% para os dados de teste e 80% para os dados de treino. O conjunto de pontos do conjunto de teste, são pertencentes à bola fechada de centro 0 e raio 0,5, ou seja, é construído um corte no parabolóide hiperbólico. Como resposta, a rede fornece o valor de  $z \approx x^2 - y^2$ .

Na Figura 1.16(a) são apresentados os dados utilizados no treinamento, enquanto na Figura 1.16(b) são mostrados os valores reconstruídos pela rede.

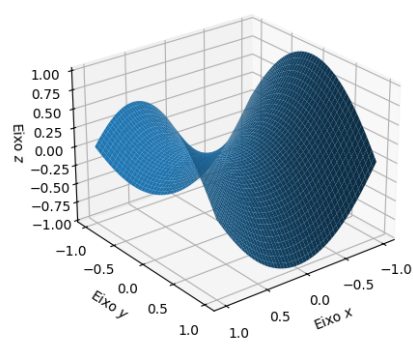
Figura 1.16: Parabolóide hiperbólico.

Superfície:  $z = x^2 - y^2$  com corte em  $x^2 + y^2 \leq 0.25$



(a)

Aproximação da Superfície:  $z = x^2 + y^2$



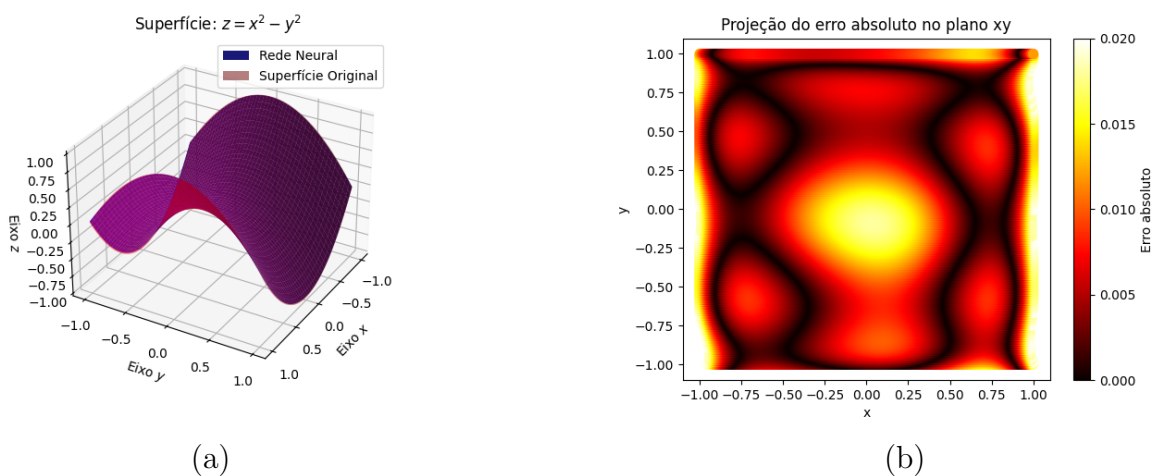
(b)

Fonte: Elaborado pelo autor.

A abordagem proposta obteve uma acurácia de 93% na reconstrução do parabolóide. O cálculo da acurácia foi realizado a partir dos dados de teste, considerando a condição  $|\hat{y} - y| \leq 0,01$ , em que  $\hat{y}$  representa a resposta fornecida pela rede e  $y$  a solução analítica. Como toda aproximação, existe um erro entre a forma analítica e a aproximada pela MLP, ao calcular a

saída para os dados  $x = 1, y = 1$  a rede responde  $-0,0192$ . Observa-se que o erro excede a precisão definida na acurácia, uma vez que, dos 6443 pontos no conjunto de teste, aproximadamente 6014 estão dentro da margem de tolerância, resultando na acurácia de 93%. Caso a precisão fosse ampliada para 0,02, a acurácia alcançaria 99%, podendo apresentar erro superior a 0,02. Na Figura 1.17(a) são apresentados o gráfico da função  $z$  e a superfície aproximada pela rede. A projeção da superfície do erro absoluto, dado por:  $E = |\hat{y} - y| \leq 0,02$ , em que  $\hat{y}$  representa a resposta fornecida pela rede e  $y$  a solução analítica é exibida na Figura 1.17(b), em que os valores de “ $E$ ” representados pelas cores mais claras estão próximos de 0,02 e os mais escuros, próximos de 0.

Figura 1.17: Gráfico da superfície e aproximação da superfície obtida pela MLP e a projeção da superfície do erro absoluto no plano  $xy$ .



Fonte: Elaborado pelo autor.

É possível observar que na Figura 1.17(b) a região de maior erro está concentrada na região do corte, ou seja, onde os dados de treinamento foram retirados.

## 1.3 Rede Neural para Análise de Imagens

### 1.3.1 Córtex Visual Humano e Visão Computacional

O córtex visual humano é a parte do cérebro responsável pelo processamento das informações visuais recebidas pelos olhos, que está localizado na parte posterior do cérebro, na região occipital (STANFIELD, 2014). O córtex visual é organizado em várias áreas, cada uma especializada em processar diferentes aspectos da visão, como cor, forma, movimento e profundidade. Essas áreas trabalham em conjunto para interpretar e compreender o mundo visual ao nosso redor. A estrutura do córtex visual é altamente complexa, com bilhões de neurônios interconectados que formam redes neurais intrincadas. Essas redes são capazes de aprender e adaptar-se com base nas experiências visuais, permitindo-nos reconhecer objetos, rostos e padrões visuais de maneira eficiente. Uma característica importante do córtex visual é a presença de campos receptivos,

que são regiões específicas do campo visual onde a estimulação de uma unidade resulta em uma resposta. Esses campos receptivos são organizados de forma hierárquica, com neurônios em níveis mais baixos respondendo a características simples, como bordas e texturas, enquanto neurônios em níveis mais altos respondem a características mais complexas, como formas e objetos completos. É por meio desse mecanismo e também pela capacidade de aprendizagem que o ser humano é capaz de reconhecer e interpretar o mundo visual ao seu redor.

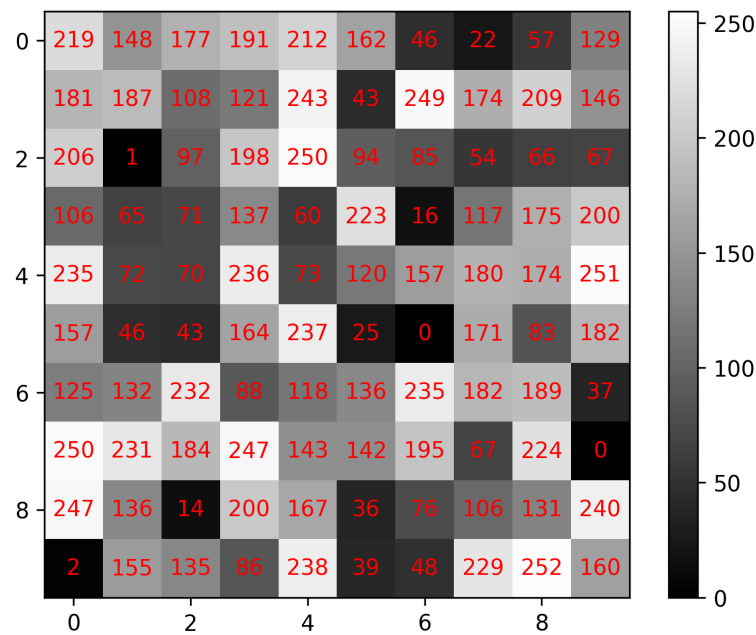
Traduzir a visão para uma representação digital é um processo extremamente complexo. Em parte, porque é um problema inverso, no qual busca-se recuperar algumas variáveis desconhecidas a partir de informações insuficientes para especificar completamente a solução. Portanto, deve-se recorrer a modelos baseados na física, modelos probabilísticos ou aprendizagem de máquina com grandes conjuntos de exemplos para resolver ambiguidades entre possíveis soluções. Entretanto, modelar o mundo visual em toda a sua rica complexidade é muito mais difícil do que, por exemplo, modelar o trato vocal que produz sons da fala (SZELISKI, 2010).

É surpreendente que humanos e animais façam isso com tanta facilidade, enquanto algoritmos de visão computacional são tão propensos a erros. Pessoas que nunca trabalharam na área frequentemente subestimam a dificuldade do problema. Essa percepção equivocada de que a visão deveria ser fácil remonta aos primeiros dias da inteligência artificial, quando se acreditava inicialmente que as partes cognitivas (prova lógica e planejamento) da inteligência eram intrinsecamente mais difíceis do que os componentes perceptivos (MÜLLER, 2008).

Uma imagem virtual é uma representação visual de um objeto, cena ou conceito, composta por uma matriz de pixels que contém informações sobre cor e intensidade. Cada pixel é uma célula na matriz da imagem que possui um valor específico. As imagens podem ser em escala de cinza, em que cada pixel tem um valor de intensidade entre 0 (preto) e 255 (branco), como mostrado na Figura 1.18, ou em cores, em que cada pixel é representado por uma combinação de valores para as cores primárias: vermelho, verde e azul, do inglês *Red*, *Green*, *Blue* (RGB). Uma imagem colorida pode ser interpretada como a sobreposição de três matrizes, cada matriz representando uma das cores primárias, essas matrizes são chamadas de canais, por exemplo, uma imagem colorida de  $10 \times 10$  pixels seria representada por uma matriz de  $10 \times 10 \times 3$ , onde cada elemento da matriz contém os valores de intensidade para as cores vermelha, verde e azul. Entretanto, nem sempre os canais representam uma matriz de cor, podendo representar outras informações, como a profundidade de um objeto em uma cena ou uma característica específica da imagem. Nos próximos capítulos, as dimensões de uma imagem serão representadas por  $B \times H \times W \times C$ , em que  $B$  é o tamanho do lote, sendo este utilizado para calcular o SGD, apresentado na Subseção 1.2.2,  $H$  é a altura da imagem,  $W$  é a largura da imagem e  $C$  é o número de canais. Quando  $B$  ou  $C$  são iguais a 1, a representação pode aparecer como  $H \times W$ .

Na Figura 1.18 a imagem tem dimensões  $1 \times 10 \times 10 \times 1$ , ou seja,  $B = 1$ ,  $H = 10$ ,  $W = 10$  e  $C = 1$ .

Figura 1.18: Exemplo de uma imagem/matriz  $1 \times 10 \times 10 \times 1$  em escala de cinza, em que cada pixel tem um valor de intensidade entre 0 (preto) e 255 (branco).



Fonte: Elaborado pelo autor.

### 1.3.2 Redes Neurais Convolucionais

As redes neurais convolucionais, do inglês *Convolutional Neural Networks* (CNNs), são uma classe especial de redes neurais projetadas para processar dados com uma estrutura de grade e que tenham uma relação de vizinhança entre os elementos, como imagens. Foram baseadas do Cortex Visual Humano (TREVISAN, 2021), estas são amplamente utilizadas em tarefas de visão computacional, como reconhecimento de objetos, detecção de faces e segmentação de imagens. Segmentação de imagens é uma modificação feita na área de visão computacional que envolve a divisão de uma imagem em regiões ou segmentos distintos, cada um representando um objeto ou parte de um objeto. O objetivo da segmentação é atribuir uma etiqueta a cada pixel da imagem, indicando a qual classe ou categoria pertence. As CNNs são inspiradas na organização do córtex visual, onde diferentes regiões do cérebro são responsáveis por processar diferentes partes do campo visual.

### 1.3.3 Convolução

**Exemplo 1.3.** Supondo que na tentativa de localizar uma nave com um sensor a laser, este sensor retorna uma única saída  $x(t)$ , a posição da nave no tempo  $t$ , que são valores reais, ou seja, é possível adquirir diferentes leituras do sensor em qualquer instante de tempo. Supondo que o sensor seja ruidoso, para obter pouco ruído da estimativa da posição da nave é possível fazer várias medições e calcular a média destas, entretanto, medições mais recentes são mais relevantes, então, pode-se fazer uma média ponderada que dê pesos às medições mais recentes (GOODFELLOW; BENGIO; COURVILLE, 2016). Seja  $w(a)$  a função que determina os pesos

atribuídos às diferentes medições. Ao aplicar  $w(a)$  à  $x(t)$ , para todo  $t$ , é obtida a função  $s(t)$ , que é a estimativa suavizada da posição da nave, ou seja:

$$s(t) = \int_{-\infty}^{\infty} x(a)w(t-a)da. \quad (1.22)$$

Essa operação é chamada de convolução (SOVIERZOSKI, 2010). A operação de convolução é tipicamente denotada com um asterístico:

$$s(t) = (x * w)(t).$$

No Exemplo 1.3, para que a saída seja uma média ponderada das medições,  $w$  deve ser não negativo e satisfazer  $\int_{-\infty}^{\infty} w(\tau) d\tau = 1$ . Assim,  $w$  pode ser interpretado como uma função densidade de probabilidade para o atraso  $\tau = t - a$ . Além disso, para que a estimativa no instante  $t$  dependa somente de medições já realizadas, deve-se ter  $w(\tau) = 0$  para  $\tau < 0$ , ou, equivalentemente,  $w(t-a) = 0$  para  $a > t$ . Para privilegiar as medições mais recentes, é usual escolher  $w$  decrescente para  $\tau \geq 0$ . Estas restrições são particulares do exemplo, em geral, convolução é definida para quaisquer funções e integrais.

Na terminologia da rede convolucional, o primeiro termo ( $x$ ) é frequentemente chamado de *input* (entrada), o segundo termo ( $w$ ) é chamado de *kernel* (núcleo) ou filtro, e o resultado ( $s$ ) é chamado de *feature map* (mapa de características).

A operação de convolução é comumente aplicada a dados provenientes de domínios discretos (dados de computador). Se  $x$  e  $w$  são definidas somente para valores inteiros de  $t$ , a convolução discreta é definida por:

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a), \quad (1.23)$$

em muitos materiais, a convolução discreta pode ser representada por:

$$s[t] = (x * w)[t] = \sum_{a=-\infty}^{\infty} x[a]w[t-a].$$

**Exemplo 1.4.** Seja  $f(t) = e^{-t}u(t)$  e  $g(t) = u(t)$ , onde  $u(t)$  é a função limiar, equação (1.2). A convolução  $(f * g)(t)$  é dada por:

$$(f * g)(t) = \int_{-\infty}^{\infty} e^{-a}u(a)u(t-a)da.$$

Para  $t < 0$ , a função limiar  $u(t-a)$  é zero para todos os valores de  $a$ , o que resulta em  $(f * g)(t) = 0$ . Para  $t \geq 0$ , a função limiar  $u(t-a)$  é igual a 1 para  $a \leq t$  e zero para  $a > t$ . Portanto, a convolução pode ser reescrita como:

$$(f * g)(t) = \int_0^t e^{-a}da.$$

Calculando a integral, obtem-se:

$$(f * g)(t) = [-e^{-a}]_0^t = 1 - e^{-t}.$$

Portanto, a convolução entre as funções  $f(t)$  e  $g(t)$  é dada por:

$$(f * g)(t) = \begin{cases} 0, & t < 0 \\ 1 - e^{-t}, & t \geq 0 \end{cases}. \quad (1.24)$$

Intuitivamente, a função  $g$  é invertida e deslocada por  $t$  e depois multiplicada ponto a ponto com a função  $f$ , e o resultado é integrado sobre todo o domínio contínuo. Para domínios discretos, o Exemplo 1.5 mostra o processo de convolução entre duas sequências finitas.

**Exemplo 1.5.** Seja  $x[n] = \{1, 2, 3\}$  e  $w[n] = \{0, 1\}$ , em que  $x[0] = 1$ ,  $x[1] = 2$ ,  $x[2] = 3$ ,  $w[0] = 0$  e  $w[1] = 1$  e  $w[n] = x[n] = 0$  para outros valores de  $n$ . A convolução  $(x * w)[n]$  é dada por:

$$s[n] = (x * w)[n] = \sum_{a=0}^3 x[a]w[n-a].$$

Calculando os valores para cada  $n$ , obtém-se:

$$\begin{aligned} (n=0) \quad (x * w)[0] &= x[0]w[0] + x[1]w[-1] + x[2]w[-2] = 1 \cdot 0 + 2 \cdot 0 + 3 \cdot 0 = 0 \\ (n=1) \quad (x * w)[1] &= x[0]w[1] + x[1]w[0] + x[2]w[-1] = 1 \cdot 1 + 2 \cdot 0 + 3 \cdot 0 = 1 \\ (n=2) \quad (x * w)[2] &= x[0]w[2] + x[1]w[1] + x[2]w[0] = 1 \cdot 0 + 2 \cdot 1 + 3 \cdot 0 = 2 \\ (n=3) \quad (x * w)[3] &= x[0]w[3] + x[1]w[2] + x[2]w[1] = 1 \cdot 0 + 2 \cdot 0 + 3 \cdot 1 = 3 \end{aligned}$$

Portanto, a convolução entre as sequências  $x[n]$  e  $w[n]$  é dada por:

$$(x * w)[n] = \{0, 1, 2, 3\}. \quad (1.25)$$

**Proposição 1.1.** O tamanho do vetor resultante da convolução entre dois vetores de comprimento  $M$  e  $N$  é dado por  $M + N - 1$ .

*Demonstração.* Seja  $x[n] \neq 0$  para  $0 \leq n \leq M - 1$  e  $w[n] \neq 0$  para  $0 \leq n \leq N - 1$ . Duas sequências finitas de comprimentos  $M$  e  $N$  respectivamente. Então, a convolução  $(x * w)[n]$  é dada por:

$$y[n] = (x * w)[n] = \sum_{a=-\infty}^{\infty} x[a]w[n-a],$$

para que a soma seja diferente de zero, é necessário que  $x[a] \neq 0$  e  $w[n-a] \neq 0$ , ou seja:

$$0 \leq a \leq M - 1 \quad \text{e} \quad 0 \leq n - a \leq N - 1.$$

Somando  $a$  na segunda desigualdade é obtido:

$$a \leq n \leq a + N - 1.$$

Como  $a$  pode variar entre 0 e  $M - 1$ , quando  $a = 0$ , então  $n \geq 0$  e quando  $a = M - 1$ , então  $n \leq M - 1 + N - 1 = M + N - 2$ . Portanto:

$$0 \leq n \leq M + N - 2,$$

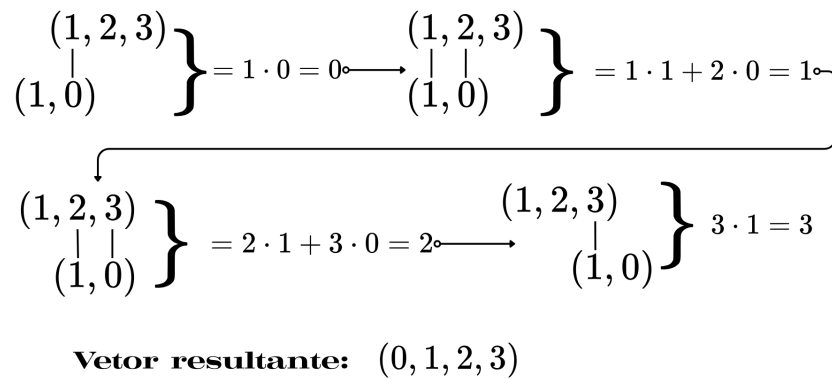
o que implica que o vetor resultante ( $y[n]$ ) tem comprimento  $M + N - 1$  □

No Exemplo 1.5, a convolução entre os vetores  $x[n]$  e  $w[n]$  resultou em um vetor de comprimento 4, que é igual a  $3 + 2 - 1 = 4$ . Para voltar ao comprimento original do vetor  $x[n]$ , pode-se fazer um truncamento do vetor resultante, ou seja, remover a mesma quantidade de elementos do início e do final do vetor resultante, o que é conhecido como convolução “*same*”, do português, mesma. Esta convolução é usada quando a dimensão do dado de entrada deve ser mantida na saída, quando isso não é necessário, a convolução pode receber outro nome, como por exemplo, convolução *valid*, utilizada quando os valores da saída são somente calculados quando o *kernel* está totalmente contido na entrada (CHOWDHURY, 2024). No Exemplo 1.5, removendo o primeiro e o último elemento do vetor resultante, obtém-se a convolução *same*:

$$(x * w)_{same}[n] = \{1, 2\}.$$

Pode-se pensar na operação de convolução discreta como o processo de multiplicação e soma, em que o kernel  $w$  vai sendo “passado” sobre o vetor  $x$  ponto a ponto, como é mostrado na Figura 1.19.

Figura 1.19: Ilustração da operação de convolução discreta.



Fonte: Elaborado pelo autor.

No contexto de redes neurais convolucionais, a convolução é aplicada a dados discretos, como

imagens representadas por matrizes de pixels. Neste caso, o sinal  $x$  é uma matriz bidimensional que representa a imagem de entrada e o *kernel*  $w$  é uma matriz menor que atua como um filtro para extrair características específicas da imagem. A operação de convolução em dados discretos é realizada deslizando o *kernel* sobre a imagem e calculando a soma ponderada dos valores dos pixels cobertos pelo *kernel*. A saída da convolução é uma nova matriz, chamada de Mapa de Características, que destaca as características extraídas pela aplicação do *kernel*. No caso de uma convolução bidimensional discreta, a operação é definida como:

$$S[i, j] = (X * W)[i, j] = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} X[m, n]W[i - m, j - n], \quad (1.26)$$

convolução é uma operação comutativa, ou seja:

$$S[i, j] = (X * W)[i, j] = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} X[i - m, j - n]W[m, n],$$

considerando que  $X$  é a matriz da imagem de entrada,  $W$  é a matriz do kernel e  $S$  é o mapa de características resultante.

**Exemplo 1.6.** *Considere a convolução de duas matrizes, em que a matriz de entrada*

$$X = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

*é convoluída com o kernel*

$$W = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

*para produzir a saída  $S$ . Então, como nos exemplos anteriores, é necessário inverter o kernel  $W$  e depois deslizar o kernel sobre a matriz de entrada  $X$ . Neste exemplo, a convolução é a “same”.*

$$W \rightarrow \hat{W} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix},$$

$\hat{W} = J_i W J_i$  em que  $J_i$  é uma matriz de permutação que realiza a rotação de 180 graus do kernel original  $W$ , ou seja:

$$J_i = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}.$$

*Para facilitar o cálculo da convolução, pode-se separar  $X$  com blocos de  $2 \times 2$  (dimensão do kernel) e calcular a convolução para cada bloco. Para o bloco superior esquerdo de  $X$  até o*

bloco inferior direito, temos:

$$\begin{aligned}
 S(0,0) &= \begin{bmatrix} 1 & 2 \\ 4 & 5 \end{bmatrix} \odot \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} = (1)(-1) + (2)(0) + (4)(0) + (5)(1) = 4, \\
 S(0,1) &= \begin{bmatrix} 2 & 3 \\ 5 & 6 \end{bmatrix} \odot \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} = (2)(-1) + (3)(0) + (5)(0) + (6)(1) = 4, \\
 S(1,0) &= \begin{bmatrix} 4 & 5 \\ 7 & 8 \end{bmatrix} \odot \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} = (4)(-1) + (5)(0) + (7)(0) + (8)(1) = 4, \\
 S(1,1) &= \begin{bmatrix} 5 & 6 \\ 8 & 9 \end{bmatrix} \odot \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} = (5)(-1) + (6)(0) + (8)(0) + (9)(1) = 4,
 \end{aligned}$$

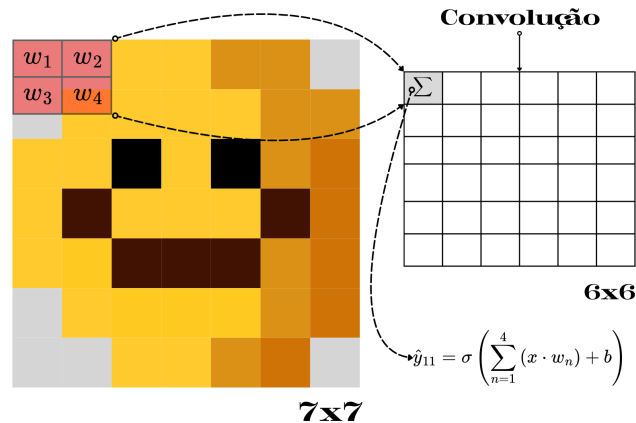
em que a operação  $\odot$  significa a multiplicação de cada elemento correspondente das matrizes e a soma dos resultados. Portanto, a saída da convolução é dada por:

$$S = \begin{bmatrix} 4 & 4 \\ 4 & 4 \end{bmatrix}. \quad (1.27)$$

Observa-se que a saída da convolução é uma matriz de  $2 \times 2$ . O cálculo da dimensão da matriz de saída é feito da seguinte forma:  $Largura_S = largura_X - largura_W + 1 = 3 - 2 + 1 = 2$  e  $Altura_S = altura_X - altura_W + 1 = 3 - 2 + 1 = 2$ .

Na Figura 1.20 é ilustrado um exemplo de convolução bidimensional, em que uma matriz de entrada  $X$  (matriz  $7 \times 7$ ) é convoluída com um *kernel*  $W$  (matriz  $2 \times 2$ ) para produzir a saída  $S$  (matriz  $6 \times 6$ ).  $W$  é deslizado sobre a matriz de entrada, e em cada posição, a soma ponderada dos valores dos pixels cobertos pelo *kernel* é calculada e o resultado é aplicado em uma função de ativação ( $\sigma$ ) para gerar o valor correspondente na matriz de saída, assim como no Exemplo 1.6. No caso da Figura 1.20, é utilizado a convolução válida, ou seja, a saída é gerada apenas para as posições onde o *kernel* pode ser completamente aplicado à matriz de entrada sem ultrapassar seus limites.

Figura 1.20: Exemplo de convolução bidimensional.



Fonte: Elaborado pelo autor.

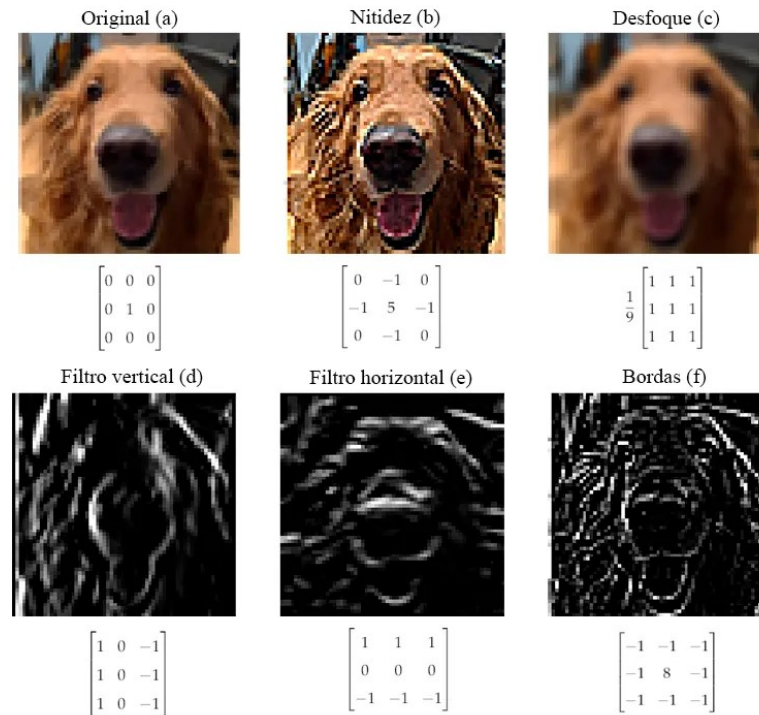
O cálculo da largura e da altura da imagem de saída pode ser feito utilizando as seguintes fórmulas:

$$w_{out} = w_{in} - w_{ker} + 1, \quad (1.28)$$

$$h_{out} = h_{in} - h_{ker} + 1, \quad (1.29)$$

em que  $w_{out}$  e  $h_{out}$  são a largura e a altura da imagem de saída,  $w_{in}$  e  $h_{in}$  são a largura e a altura da imagem de entrada, e  $w_{ker}$  e  $h_{ker}$  são a largura e a altura do *kernel*, respectivamente. Na Figura 1.21 são mostrados diferentes aplicações de *kernels* em uma imagem. A imagem original é representada na Figura 1.21 (a), juntamente com o *kernel* identidade, que não faz modificações. Na Figura 1.21 (b) a imagem passou por um *kernel* que aumenta a nitidez reforçando o pixel central e aumentando a diferença dele em relação à sua vizinhança. Na Figura 1.21 (c) foi usado um *kernel* de desfoque, que substitui um pixel pela média dele com seus vizinhos. A Figura 1.21 (d) é o resultado de aplicar na imagem original em escala de cinza um *kernel* que reforça suas linhas verticais, enquanto na Figura 1.21 (e) reforça suas linhas horizontais. O *kernel* na Figura 1.21 (f) reforça os contornos (bordas) da imagem.

Figura 1.21: Aplicações de diferentes *kernels* em uma imagem.



Fonte: Adaptado de Trevisan (2021).

Normalmente, os *kernels* são muito menores do que as imagens (geralmente  $3 \times 3$ ), e através da operação de convolução um *kernel* é utilizado para processar toda uma imagem, portanto, diz-se que os parâmetros são compartilhados (pesos e viés). Por conta dessa representação reduzida, cada camada da rede convolucional deve aprender uma quantidade menor de parâmetros. Em CNNs, considera-se a convolução como uma camada da rede, e cada camada possui múltiplos *kernels*, produzindo a mesma quantidade de mapas de atributos, cada um res-

ponsável por extrair diferentes características da imagem de entrada. Durante o treinamento da rede, os valores dos *kernels* são ajustados para otimizar a capacidade da rede de reconhecer padrões específicos nas imagens. Por exemplo, uma camada convolucional com 3 *kernels* que recebe como entrada uma imagem em escala de cinza de dimensões  $(w_{in} \times h_{in} \times 1)$  produzirá uma saída com dimensões  $(w_{out} \times h_{out} \times 3)$ , onde  $w_{out}$  e  $h_{out}$  são calculados conforme as fórmulas apresentadas em (1.28) e (1.29). A saída não é mais uma imagem comum, mas sim um conjunto de três mapas de características, cada um representando a resposta da imagem de entrada a um dos três *kernels* aplicados. Encadeando os mapas de atributos, a rede pode aprender representações hierárquicas das características presentes nas imagens, desde bordas e texturas simples até formas e objetos mais complexos.

### 1.3.4 Correlação Cruzada

As redes neurais convolucionais geralmente não utilizam a definição tradicional de convolução, mas sim uma operação chamada correlação cruzada, definida como:

$$S[i, j] = (X * W)[i, j] = \sum_m \sum_n X[i + m, j + n] W[m, n], \quad (1.30)$$

em que  $X$  é a matriz da imagem de entrada,  $W$  é a matriz do *kernel* e  $S$  é o mapa de características resultante (GOODFELLOW; BENGIO; COURVILLE, 2016). A mudança ocorre porque no processo de treinamento os valores do *kernel* não são definidos manualmente, mas sim aprendidos por meio dos processos de otimização dos pesos. Por conta disso, a rede aprende a melhor configuração do *kernel* sem a necessidade de realizar a rotação de 180 graus, sendo exatamente o que a definição de correlação cruzada descreve, já que  $W$  não depende de parâmetros negativos.

### 1.3.5 Kernel Multicanais

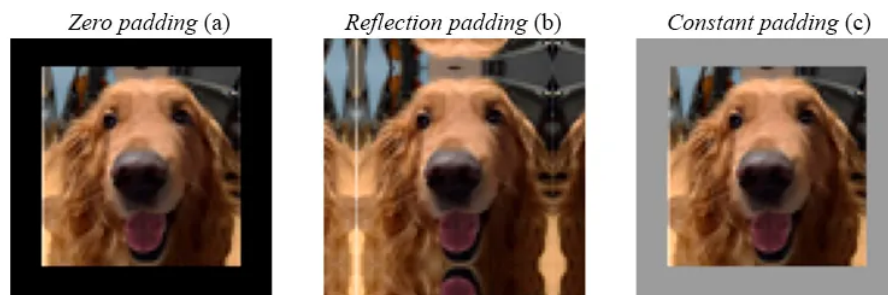
Em uma imagem com mais de um canal, como no caso de uma imagem RGB, a convolução acontece por meio de um *kernel* com a mesma quantidade de canais. Por exemplo, considere uma imagem RGB de dimensões  $(w_{in} \times h_{in} \times 3)$ , então as dimensões do *kernel* são  $(k_w \times k_h \times 3)$ . A convolução é realizada separadamente em cada canal da imagem, utilizando o respectivo canal do *kernel*. Após a aplicação do *kernel* em cada canal, os resultados são somados ou é aplicada uma média para produzir um único mapa de características. Portanto, a saída da convolução tem dimensões  $(w_{out} \times h_{out} \times 1)$ , onde  $w_{out}$  e  $h_{out}$  são calculados conforme as equações apresentadas em (1.28) e (1.29).

### 1.3.6 Padding

Durante a aplicação da convolução, é comum que a dimensão espacial (largura e altura) da imagem de saída seja menor do que a da imagem de entrada. A dimensão é menor porque o *kernel* não pode ser aplicado nas bordas da imagem sem ultrapassar seus limites. Para mitigar

essa redução de dimensão, uma técnica chamada *padding*, em português, preenchimento, é frequentemente utilizada (O'SHEA; NASH, 2015). O *padding* envolve adicionar uma borda de pixels ao redor da imagem original antes de aplicar a convolução. Esses pixels adicionais podem ser preenchidos com zeros (*zero padding*) ou com valores replicados dos pixels mais próximos da borda da imagem original, (*reflection padding*). O uso de *padding* permite que o *kernel* seja aplicado em todas as regiões da imagem, incluindo as bordas, preservando assim as dimensões espaciais da imagem de entrada na saída. Na Figura 1.22 é ilustrada a aplicação de diferentes tipos de *padding* em uma imagem.

Figura 1.22: Aplicação de *padding* em uma imagem. Em (a) a imagem original passou por zero *padding*. Em (b) a imagem original passou por *reflection padding*. A imagem em (c) passou por um *padding* constante, semelhante em (a), mas com valor diferente de zero.



Fonte: Adaptado de Trevisan (2021).

O cálculo da largura e da altura da imagem de saída com *padding* pode ser ajustado conforme as seguintes fórmulas:

$$\begin{aligned} w_{out} &= w_{in} - w_{kernel} + 2w_{pad} + 1, \\ h_{out} &= h_{in} - h_{kernel} + 2h_{pad} + 1, \end{aligned} \tag{1.31}$$

em que  $w_{pad}$  e  $h_{pad}$  são a quantidade de pixels adicionados como *padding* na largura e na altura, respectivamente.

### 1.3.7 *Stride*

O *stride*, em português, passo, é um parâmetro utilizado nas operações de convolução em redes neurais convolucionais, que determina o número de pixels que o *kernel* se desloca ao longo da imagem de entrada. Em outras palavras, o *stride* define o quanto o *kernel* avança a cada aplicação da operação. Um *stride* de 1 significa que o *kernel* se mova um pixel por vez, enquanto um *stride* maior que 1 faz com que o *kernel* pule pixels, resultando em uma redução mais rápida das dimensões espaciais da imagem. O uso do *stride* afeta diretamente a dimensão da saída gerada pela operação de convolução. O cálculo da largura e da altura da imagem de saída com

*stride* pode ser ajustado conforme as seguintes fórmulas:

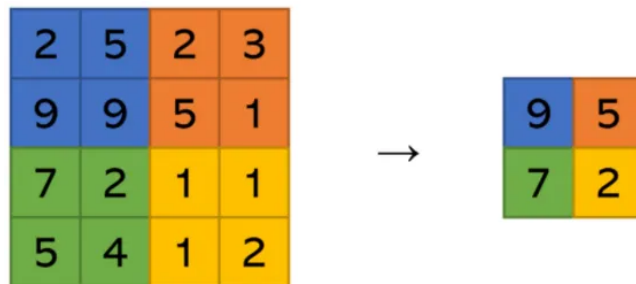
$$\begin{aligned} w_{out} &= \frac{w_{in} - w_{kernel} + 2w_{pad}}{s_w} + 1, \\ h_{out} &= \frac{h_{in} - h_{kernel} + 2h_{pad}}{s_h} + 1, \end{aligned} \quad (1.32)$$

em que  $s_w$  e  $s_h$  são os valores do *stride* na largura e na altura, respectivamente.

### 1.3.8 Pooling

O *pooling*, em português, agrupamento, é uma operação utilizada em redes neurais convolucionais para reduzir as dimensões espaciais (largura e altura) dos mapas de características, mantendo as informações mais relevantes. Essa redução ajuda a diminuir a complexidade computacional da rede. Existem diferentes tipos de operações de *pooling*, sendo as mais comuns o *Max pooling* e o *Average pooling* (Agrupamento Médio, técnica que leva em consideração a média dos valores dos pixels em cada bloco coberto pelo *kernel* de *pooling*). A técnica de *pooling* substitui agrupamentos de pixels por um valor que os represente. Na Figura 1.23 é ilustrado o funcionamento do *Max pooling* de dimensão  $2 \times 2$ , ou seja, os pixels são agrupados em quadrados de dimensões  $2 \times 2$  e são substituídos pelo maior valor de intensidade de pixel do grupo. Em CNNs o *stride* do *pooling* é geralmente igual a dimensão do *kernel* de *pooling*, o que resulta em uma redução de dimensão espacial pela metade a cada aplicação de *pooling*. Por exemplo, um mapa de características de dimensões  $8 \times 8$  aplicado a um *Max pooling* com *kernel* de dimensões  $2 \times 2$  e *stride* de 2 resultará em um mapa de características reduzido para dimensões  $4 \times 4$ .

Figura 1.23: Exemplo de *Max pooling* com dimensões  $2 \times 2$ .



Fonte: Adaptado de Trevisan (2021).

O *Max pooling* pode ser implementado utilizando o Algoritmo 3, em que  $X$  é a matriz de entrada,  $k$  é a dimensão do *kernel* de *pooling* e  $Y$  é a matriz de saída resultante do *pooling*. O algoritmo percorre a matriz de entrada em blocos de dimensões  $k \times k$ , calculando o valor máximo dentro de cada bloco e atribuindo esse valor à posição correspondente na matriz de saída  $Y$ . O resultado é uma matriz reduzida que mantém as características mais importantes da imagem original.

O cálculo da largura e da altura da imagem de saída após a aplicação do *pooling*, quando o

**Entrada:**  $X$  (matriz de entrada),  $k$  (dimensão do *kernel* de *pooling*)  
**Saída:**  $Y$  (matriz de saída)  
 $w_{out} \leftarrow \lfloor w_{in}/k \rfloor$ ;  
 $h_{out} \leftarrow \lfloor h_{in}/k \rfloor$ ;  
 $Y \leftarrow$  matriz de zeros de dimensões  $w_{out} \times h_{out}$ ;  
**Para**  $i \leftarrow 0$  **Até**  $w_{out} - 1$  **Faça**  
    **Para**  $j \leftarrow 0$  **Até**  $h_{out} - 1$  **Faça**  
         $Y[i][j] \leftarrow \max(X[i * k : (i + 1) * k][j * k : (j + 1) * k])$ ;  
    **Fim**  
**Fim**

**Algoritmo 3:** *Max pooling*

*stride* é igual à dimensão do *kernel*, pode ser realizado conforme as seguintes fórmulas:

$$\begin{aligned} w_{out} &= \lfloor \frac{w_{in}}{k} \rfloor, \\ h_{out} &= \lfloor \frac{h_{in}}{k} \rfloor, \end{aligned} \tag{1.33}$$

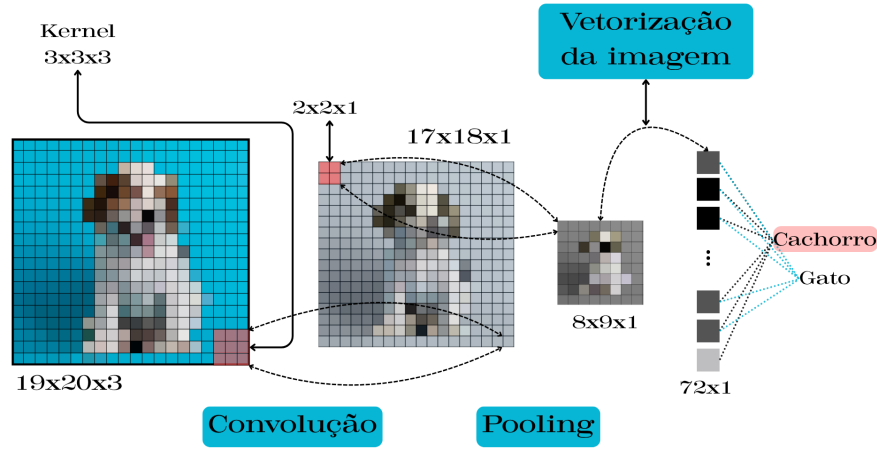
em que  $k$  é a dimensão do *kernel* de *pooling*, e  $\lfloor \cdot \rfloor$  denota a função piso, que arredonda para baixo o resultado da divisão. Como consequência imediata da aplicação do *pooling*, a informação fica mais condensada, o que pode levar a uma perda de detalhes finos na imagem. No entanto, essa perda é compensada pela capacidade de aumentar a quantidade dos mapas de característica da imagem, melhorando sua capacidade de generalização em tarefas de reconhecimento de padrões.

**Exemplo 1.7.** Na Figura 1.24 é apresentada uma arquitetura típica de uma CNN, com uma camada de convolução e uma camada de pooling. Como a imagem têm três canais (RGB) de dimensões  $19 \times 20 \times 3$  o kernel de convolução também tem três canais ( $3 \times 3 \times 3$ ), a aplicação da convolução resulta em um mapa de características com um canal ( $17 \times 18 \times 1$ ). O mapa de características é então processado por uma camada de pooling ( $2 \times 2 \times 1$ ), que reduz suas dimensões espaciais ( $8 \times 9 \times 1$ ). Após essa etapa a rede transforma o mapa de características em um vetor unidimensional ( $72 \times 1$ ), que é processado por camadas totalmente conectadas, visto na Seção 1.2 para realizar a classificação do objeto em cachorro ou gato, com a rede escolhendo a classe com maior probabilidade como resultado final (cachorro).

### 1.3.9 Convolução Transposta

A convolução transposta, também conhecida como deconvolução, é uma operação utilizada em redes neurais convolucionais para aumentar as dimensões espaciais (largura e altura) dos mapas de características. O nome sugere uma operação inversa à convolução, entretanto não é isso que acontece. Esse nome provavelmente veio da ideia de “inverter” a operação de convolução no sentido de expandir as dimensões espaciais da entrada e não diminuir. Essa operação é frequentemente empregada em tarefas de geração de imagens, em que o objetivo é reconstruir imagens a partir de representações compactas. A convolução transposta funciona de maneira

Figura 1.24: Arquitetura típica de uma Rede Neural Convolutacional com uma camada de convolução e uma camada de *pooling*.



Fonte: Elaborado pelo autor.

inversa à convolução válida. Enquanto a convolução reduz as dimensões espaciais da imagem de entrada, a convolução transposta expande essas dimensões. Este processo é alcançado gerando uma saída com dimensões calculadas da seguinte forma:

$$\begin{aligned} w_{out} &= (w_{in} - 1) \cdot s_w - 2w_{pad} + w_{kernel}, \\ h_{out} &= (h_{in} - 1) \cdot s_h - 2h_{pad} + h_{kernel}, \end{aligned} \quad (1.34)$$

em que  $w_{in}$  e  $h_{in}$  são a largura e a altura da imagem de entrada,  $s_w$  e  $s_h$  são os valores do *stride* na largura e na altura,  $w_{pad}$  e  $h_{pad}$  são a quantidade de pixels adicionados como *padding* na largura e na altura (geralmente os valores para o *padding* são zero), e  $w_{kernel}$  e  $h_{kernel}$  são a largura e a altura do *kernel*, respectivamente. Cada entrada da imagem de saída é calculada como uma soma ponderada dos valores da imagem de entrada multiplicados pelos pesos do *kernel*. A convolução transposta pode ser implementada utilizando o seguinte Algoritmo 4.

Na Figura 1.25 é exemplificada a operação de convolução transposta, em que uma matriz de entrada  $X$  (matriz  $2 \times 2$ ) é operada com um *kernel*  $W$  (matriz  $2 \times 2$ ) para produzir a saída  $Y$  (matriz  $3 \times 3$ ).

**Entrada:**  $X$  (matriz de entrada),  $W$  (*kernel*),  $s_w$  (*stride* na largura),  $s_h$  (*stride* na altura),  $w_{pad}$  (*padding* na largura),  $h_{pad}$  (*padding* na altura)

**Saída:**  $Y$  (matriz de saída)

$w_{out} \leftarrow (w_{in} - 1) \cdot s_w - 2w_{pad} + w_{kernel};$

$h_{out} \leftarrow (h_{in} - 1) \cdot s_h - 2h_{pad} + h_{kernel};$

$Y \leftarrow$  matriz de zeros de dimensão  $w_{out} \times h_{out};$

**Para**  $i \leftarrow 0$  **Até**  $w_{in} - 1$  **Faça**

**Para**  $j \leftarrow 0$  **Até**  $h_{in} - 1$  **Faça**

**Para**  $m \leftarrow 0$  **Até**  $w_{kernel} - 1$  **Faça**

**Para**  $n \leftarrow 0$  **Até**  $h_{kernel} - 1$  **Faça**

$Y[i \cdot s_w + m - w_{pad}][j \cdot s_h + n - h_{pad}] += X[i][j] \cdot W[m][n];$

**Fim**

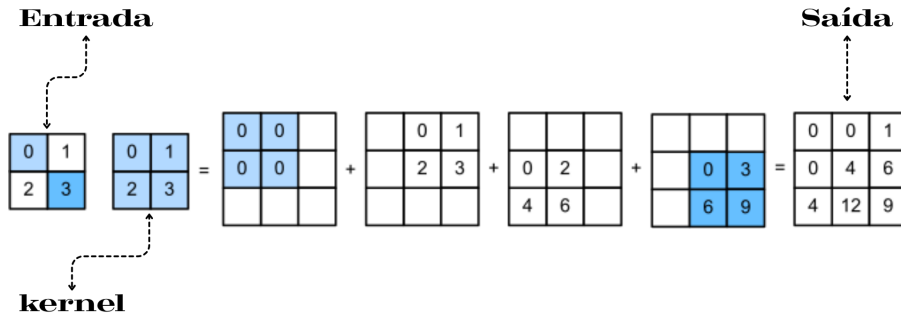
**Fim**

**Fim**

**Fim**

**Algoritmo 4:** Convolução Transposta.

Figura 1.25: Exemplo da operação de convolução transposta.



Fonte: Elaborado pelo autor.

As cores (por exemplo, azul claro) na imagem de entrada representam a posição do pixel e quais casas influencia na operação de convolução transposta.

Desta forma, este capítulo abordou desde os conceitos fundamentais das unidades até a definição formal da operação de convolução, passando por exemplos e aplicações de redes neurais. A compreensão desses conceitos é essencial para o desenvolvimento de modelos de aprendizado profundo, especialmente em tarefas relacionadas ao processamento de imagens e reconhecimento de padrões. No próximo capítulo é apresentada a teoria básica para o desenvolvimento do modelo U-Net Fuzzy.

# Capítulo 2

## Teoria dos Conjuntos Fuzzy

No cotidiano, o ser humano raramente raciocina por categorias absolutamente rígidas. Ao avaliar se um dia está quente, se uma pessoa é alta ou se uma imagem está nítida, por exemplo, normalmente não é pensado em termos de “pertence” ou “não pertence”, como estabelece a lógica clássica dos conjuntos. Em vez disso, é pensado em graus: algo pode ser um pouco quente, muito quente, relativamente alto ou razoavelmente nítido. É justamente essa forma mais gradual e flexível de interpretar a realidade que motiva a teoria dos conjuntos fuzzy. Ao permitir que um elemento pertença a um conjunto com diferentes graus de pertinência, essa abordagem se torna mais adequada para representar incertezas, imprecisões e transições suaves entre categorias. Neste capítulo, são apresentados os principais conceitos da teoria dos conjuntos fuzzy e dos sistemas baseados em regras fuzzy, que são utilizadas no desenvolvimento deste trabalho.

A teoria dos conjuntos fuzzy, introduzida por Lotfi A. Zadeh em 1965 (ZADEH, 1965), é uma extensão da teoria clássica dos conjuntos, permitindo a representação de incerteza e imprecisão. Diferentemente dos conjuntos clássicos, em que um elemento pertence ou não a um conjunto, nos conjuntos fuzzy cada elemento possui um grau de pertinência, variando entre 0 e 1. Essa abordagem é especialmente útil em análise de imagens, em que as características dos objetos podem ser ambíguas ou imprecisas, dependendo do contexto, e a classificação binária pode não ser suficiente para capturar a complexidade dos dados. As definições e propriedades dos conjuntos fuzzy presentes neste capítulo foram baseadas em Jafelice, Barros e Bassanezi (2023).

### 2.1 Conjuntos Fuzzy

Um *subconjunto fuzzy*  $A$  de um conjunto universo  $X$  é definido por uma função de pertinência  $\mu_A$  que atribui a cada elemento  $x \in X$  um valor  $\mu_A(x) \in [0, 1]$ , representando o grau de pertencimento de  $x$  ao conjunto  $A$ . Assim:

$$\mu_A : X \rightarrow [0, 1].$$

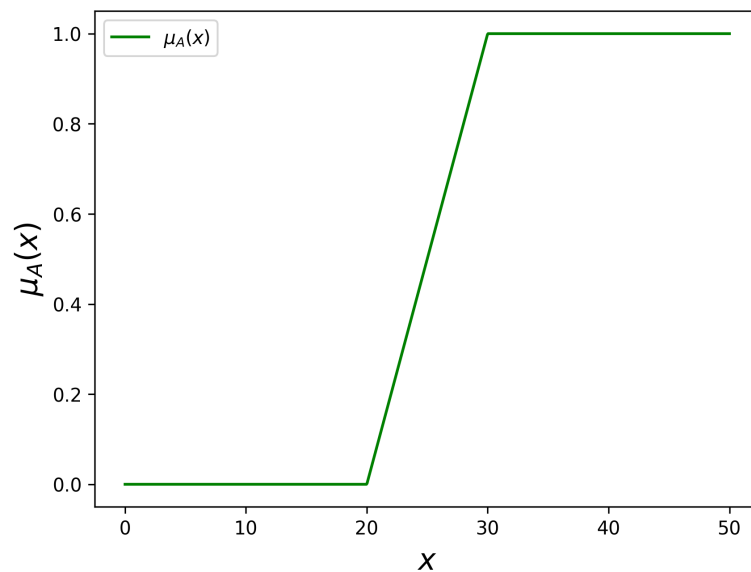
O valor  $\mu_A(x) = 0$  indica que  $x$  não pertence ao conjunto  $A$ , enquanto  $\mu_A(x) = 1$  indica que  $x$  pertence completamente ao conjunto. Valores intermediários representam graus parciais de pertencimento, permitindo uma representação mais rica e flexível de conceitos vagos ou imprecisos.

**Exemplo 2.1.** Considere o conjunto universo  $X$  representando a temperatura em graus Celsius, e o conjunto fuzzy  $A$  representando a categoria “temperatura alta”. A função de pertinência  $\mu_A$  pode ser definida da seguinte forma:

$$\mu_A(x) = \begin{cases} 0, & \text{se } 0^\circ C \leq x \leq 20^\circ C \\ \frac{x-20}{10}, & \text{se } 20^\circ C < x < 30^\circ C \\ 1, & \text{se } x \geq 30^\circ C \end{cases} \quad (2.1)$$

Neste exemplo, temperaturas de  $20^\circ C$  ou menos não são consideradas altas ( $\mu_A(x) = 0$ ), enquanto temperaturas de  $30^\circ C$  ou mais são consideradas completamente altas ( $\mu_A(x) = 1$ ). Temperaturas entre  $20^\circ C$  e  $30^\circ C$  têm um grau de pertinência que varia linearmente entre 0 e 1, refletindo a natureza gradual do conceito de temperatura alta. Na Figura 2.1 é ilustrado o gráfico da função de pertinência  $\mu_A(x)$ .

Figura 2.1: Função de pertinência  $\mu_A(x)$  para o conjunto fuzzy “temperatura alta”.



Pode-se observar que a função de pertinência  $\mu_A(x)$  é uma função crescente para valores entre  $20^\circ C$  e  $30^\circ C$ , refletindo a ideia de que à medida que a temperatura aumenta, o grau de pertencimento ao conjunto fuzzy “temperatura alta” também aumenta.

### 2.1.1 Operações entre Conjuntos Fuzzy

Os conceitos e definições que sucedem este capítulo são essenciais para a compreensão da U-Net Fuzzy, desenvolvida posteriormente. As operações entre conjuntos fuzzy, como união,

interseção e complemento, são definidas de maneira a preservar a estrutura fuzzy. A união de dois conjuntos clássicos  $A$  e  $B$  é dada por:

$$A \cup B = \{x \in U; x \in A \text{ ou } x \in B\},$$

enquanto a interseção é definida por:

$$A \cap B = \{x \in U; x \in A \text{ e } x \in B\}.$$

O complemento de um conjunto clássico  $A$  é dado por:

$$\bar{A} = \{x \in U; x \notin A\}.$$

Essas três operações têm respectivamente as funções características,  $\forall x \in X$

$$u_{A \cup B}(x) = \max(u_A(x), u_B(x)), \quad u_{A \cap B}(x) = \min(u_A(x), u_B(x)), \quad u_{\bar{A}}(x) = 1 - u_A(x).$$

**Exemplo 2.2.** *Seja  $A$  um conjunto clássico em um conjunto universo  $X$ . A função característica  $u_A$  de  $A$  é dada por:*

$$u_A(x) = \begin{cases} 1, & \text{se } x \in A \\ 0, & \text{se } x \notin A \end{cases}.$$

Assim, se  $A = \{1, 2, 3\}$  e  $B = \{3, 4, 5\}$ , com  $u_A = u_B$ , é obtida a Tabela 2.1:

Tabela 2.1: Operações entre conjuntos clássicos.

| $x$ | $u_A(x)$ | $u_B(x)$ | $u_{A \cup B}(x)$ | $u_{A \cap B}(x)$ | $u_{\bar{A}}(x)$ |
|-----|----------|----------|-------------------|-------------------|------------------|
| 1   | 1        | 0        | 1                 | 0                 | 0                |
| 2   | 1        | 0        | 1                 | 0                 | 0                |
| 3   | 1        | 1        | 1                 | 1                 | 0                |
| 4   | 0        | 1        | 1                 | 0                 | 1                |
| 5   | 0        | 1        | 1                 | 0                 | 1                |

A união, interseção e complemento de conjuntos fuzzy são definidas de maneira análoga, sendo uma extensão das operações clássicas, utilizando as funções de pertinência.

**Definição 2.1.** *Sejam  $A$  e  $B$  conjuntos fuzzy em um conjunto universo  $X$ . A união, interseção e complemento de  $A$  e  $B$ , denotadas por  $A \cup B$ ,  $A \cap B$  e  $A'$ , respectivamente, são as funções de pertinência, presentes na Figura 2.2, definidas por:*

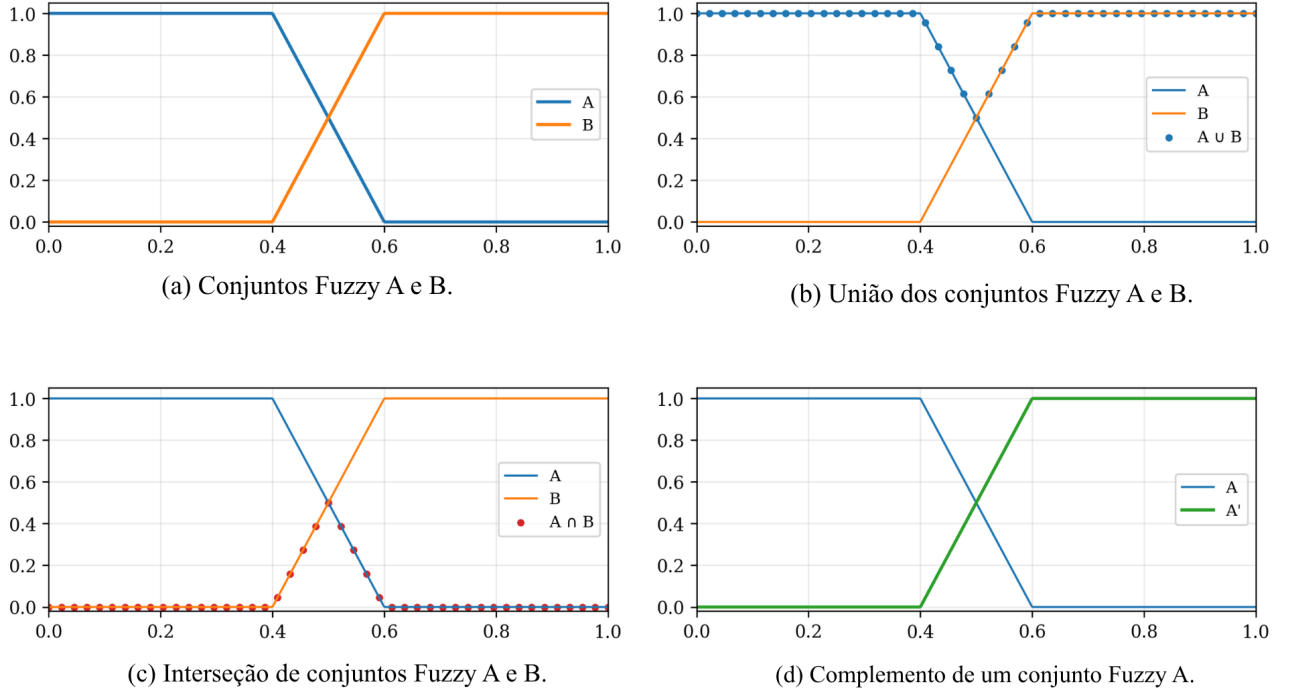
$$\mu_{A \cup B}(x) = \max(\mu_A(x), \mu_B(x)) \quad \text{para todo } x \in X,$$

$$\mu_{A \cap B}(x) = \min(\mu_A(x), \mu_B(x)) \quad \text{para todo } x \in X,$$

$$\mu_{\bar{A}}(x) = 1 - \mu_A(x) \quad \text{para todo } x \in X.$$

Observa-se que os valores obtidos para o Exemplo 2.2,  $u_{A \cap B}(x) = \mu_{A \cap B}(x)$ ,  $u_{A \cup B}(x) = \mu_{A \cup B}(x)$  e  $u_{\bar{A}}(x) = \mu_{\bar{A}}(x)$ . Na Figura 2.2 são ilustrados os gráficos das funções de pertinência para a união, interseção e complemento dos conjuntos fuzzy  $A$  e  $B$ .

Figura 2.2: Operações entre conjuntos fuzzy.



Fonte: Adaptado de Jafelice, Barros e Bassanezi (2023).

### 2.1.2 Normas Triangulares

As normas triangulares são generalizações dos operadores união e intersecção.

**Definição 2.2.** Uma co-norma triangular (*s-norma*) é uma operação binária  $S : [0, 1] \times [0, 1] \rightarrow [0, 1]$  que satisfaz as seguintes propriedades:

- *Comutatividade:*  $xSy = ySx$ ;
- *Associatividade:*  $xS(ySz) = (xSy)Sz$ ;
- *Monotonicidade:* Se  $x \leq x'$  e  $y \leq y'$ , então  $xSy \leq x'Sy'$ ;
- *Condição de fronteira:*  $xS0 = x$  e  $xS1 = 1$ .

**Exemplo 2.3.** União Padrão:  $S : [0, 1] \times [0, 1] \rightarrow [0, 1]$  com  $xSy = \max(x, y)$  é uma co-norma triangular.

De fato,  $\max(x, y) = \max(y, x)$ ,  $\max(x, \max(y, z)) = \max(\max(x, y), z)$ ,  $\max(x, y) \leq \max(x', y')$  se  $x \leq x'$  e  $y \leq y'$ ,  $\max(x, 0) = x$  e  $\max(x, 1) = 1$  para todo  $x \in [0, 1]$ .

**Definição 2.3.** Uma norma triangular (*t-norma*) é uma operação binária  $T : [0, 1] \times [0, 1] \rightarrow [0, 1]$  que satisfaz as seguintes propriedades:

- *Comutatividade:*  $xTy = yTx$ ;
- *Associatividade:*  $xT(yTz) = (xTy)Tz$ ;
- *Monotonicidade:* Se  $x \leq x'$  e  $y \leq y'$ , então  $xTy \leq x'Ty'$ ;
- *Condição de fronteira:*  $0Tx = 0$  e  $1Tx = x$ .

**Exemplo 2.4.** *Intersecção Padrão:*  $T : [0, 1] \times [0, 1] \rightarrow [0, 1]$  com  $xTy = \min(x, y)$  é uma norma triangular.

De fato, observa-se que  $\min(x, y) = \min(y, x)$ ,  $\min(x, \min(y, z)) = \min(\min(x, y), z)$ ,  $\min(x, y) \leq \min(x', y')$  se  $x \leq x'$  e  $y \leq y'$ ,  $\min(0, x) = 0$  e  $\min(1, x) = x$  para todo  $x \in [0, 1]$ .

### 2.1.3 Níveis de um Conjunto Fuzzy

**Definição 2.4.** Sejam  $A$  um conjunto fuzzy e  $\alpha \in (0, 1]$ . O  $\alpha$ -nível de  $A$ , denotado por  $[A]^\alpha$ , é o conjunto clássico definido por:

$$[A]^\alpha = \{x \in X \mid \mu_A(x) \geq \alpha\}.$$

**Definição 2.5.** O suporte de um conjunto fuzzy  $A$  são todos os elementos de  $X$  que têm grau de pertinência maior que zero em  $A$ , ou seja:

$$\text{supp}(A) = \{x \in X \mid \mu_A(x) > 0\}.$$

**Definição 2.6.** O nível zero de um conjunto fuzzy  $A$  do universo  $X$ , em que  $X$  é um espaço topológico, é o fecho topológico do suporte de  $A$ , ou seja:

$$[A]^0 = \overline{\text{supp}(A)}.$$

**Exemplo 2.5.** Seja  $A$  o conjunto fuzzy representando a categoria “temperatura alta” do Exemplo 2.1, dada pela função de pertinência (2.1).

O  $\alpha$ -nível de  $A$  para  $\alpha = 0,5$  é dado por:

$$[A]^{0,5} = \{x \in \mathbb{R} \mid \mu_A(x) \geq 0,5\}.$$

Neste caso, tem-se:

$$\mu_A(x) \geq 0,5 \implies \frac{x - 20}{10} \geq 0,5 \implies x \geq 25^\circ\text{C}.$$

Portanto, o  $\alpha$ -nível de  $A$  para  $\alpha = 0,5$  é o intervalo  $[25^\circ\text{C}, \infty)$ , representando a “temperatura alta” com um grau de pertinência de pelo menos 0,5.

O suporte de  $A$  é dado por:

$$\mu_A(x) > 0 \implies \frac{x - 20}{10} > 0 \implies x > 20^\circ C.$$

Portanto, o suporte de  $A$  é o intervalo  $(20^\circ C, \infty)$ , representando a “temperatura alta” que tem um grau de pertinência maior que zero em  $A$ .

### 2.1.4 Números Fuzzy

Um número fuzzy é um subconjunto fuzzy do conjunto dos números reais, caracterizado por uma função de pertinência que atribui a cada número real um grau de pertencimento. Os números fuzzy são frequentemente utilizados para representar quantidades imprecisas ou vagamente definidas.

**Definição 2.7.** Um conjunto fuzzy  $N$  é chamado de número fuzzy quando o conjunto universo, em que  $N$  é definido, é o conjunto dos números reais  $\mathbb{R}$ , e a função de pertinência  $\mu_N$  é tal que:

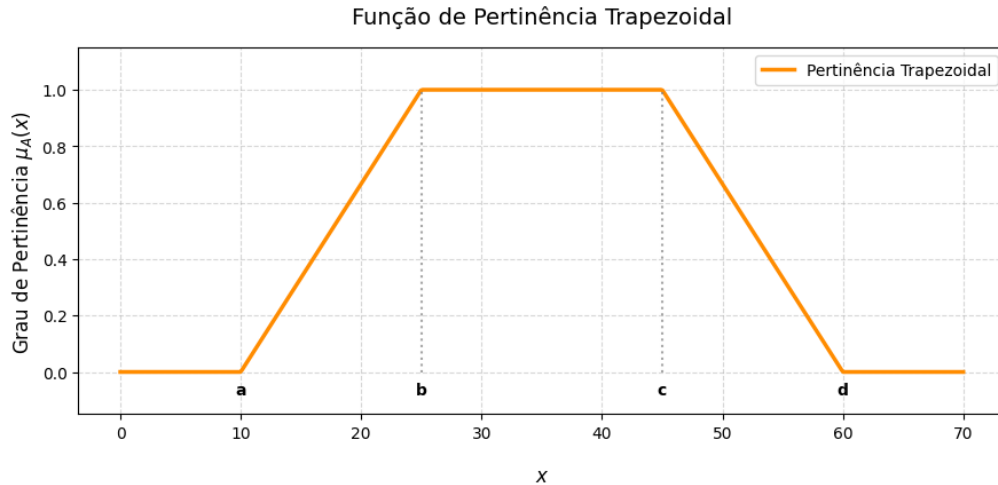
- $\mu_N(x) = 1$  para pelo menos um valor  $x$  de  $\text{supp}(N)$ ;
- $[N]^\alpha$  é um intervalo fechado para todo  $\alpha \in (0, 1]$ ;
- O suporte de  $N$  é limitado.

**Exemplo 2.6.** Um número fuzzy  $A$  é dito trapezoidal se o gráfico de sua função de pertinência tem a forma de um trapézio. A função de pertinência de um número fuzzy trapezoidal pode ser definida por quatro parâmetros  $a, b, c, d$  (com  $a \leq b \leq c \leq d$ ) da seguinte forma:

$$\mu_A(x) = \begin{cases} \frac{x-a}{b-a}, & \text{se } a < x < b \\ 1, & \text{se } b \leq x < c \\ \frac{d-x}{d-c}, & \text{se } c \leq x < d \\ 0, & \text{caso contrário} \end{cases}.$$

Neste exemplo, o número fuzzy  $A$  é representado por um trapézio definido pelos parâmetros  $a, b, c, d$ . Para  $a < x < b$ , a função de pertinência aumenta linearmente de 0 a 1, indicando um aumento gradual no grau de pertencimento. Para  $b \leq x < c$ , a função de pertinência é constante em 1, indicando que esses valores pertencem completamente ao conjunto fuzzy. Para  $c \leq x < d$ , a função de pertinência diminui linearmente de 1 a 0, indicando uma diminuição gradual no grau de pertencimento, como é mostrado na Figura 2.3.

Figura 2.3: Função de pertinência de um número fuzzy trapezoidal.



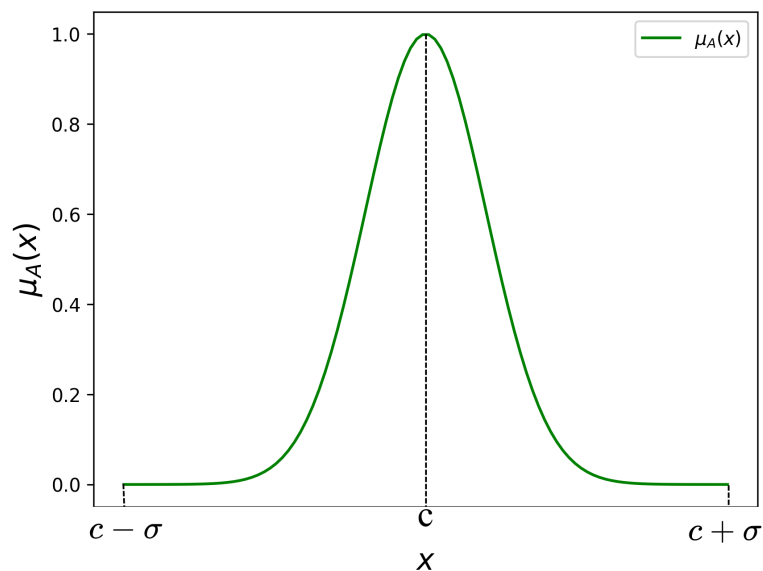
Fonte: Elaborado pelo autor.

**Exemplo 2.7.** Um número fuzzy  $A$  é dito gaussiano se o gráfico de sua função de pertinência tem a forma de uma curva gaussiana, dada por:

$$\mu_A(x) = \begin{cases} e^{-\frac{(x-c)^2}{\sigma^2}}, & \text{se } c - \sigma \leq x \leq c + \sigma \\ 0, & \text{caso contrário} \end{cases}, \quad (2.2)$$

em que  $c$  é o centro do número fuzzy e  $\sigma$  é o desvio padrão, que controla a largura da curva. Na Figura 2.4 é ilustrado o gráfico da função de pertinência de um número fuzzy gaussiano.

Figura 2.4: Função de pertinência de um número fuzzy gaussiano.



Fonte: Elaborado pelo autor.

O número fuzzy gaussiano é caracterizado por uma função de pertinência suave e contínua, que atinge seu valor máximo em  $x = c$  e decresce à medida que  $x$  se afasta de  $c$ . Observa-se que

o gráfico da função é limitado para o intervalo  $[c - \sigma, c + \sigma]$ , para que seja satisfeita a condição de suporte limitado.

## 2.2 Sistema Baseado em Regras Fuzzy

Para que seja possível mesclar os conceitos de redes neurais com a teoria fuzzy, é necessário compreender a estrutura e o funcionamento de um Sistema Baseado em Regras Fuzzy (SBRF). Um SBRF é um sistema que utiliza a teoria dos conjuntos fuzzy para produzir saídas a partir de entradas dadas por conjuntos fuzzy (JAFELICE; BARROS; BASSANEZI, 2023). Tais sistemas foram desenvolvidos com o objetivo de imitar o comportamento humano considerando que o conhecimento e as ações humanas são apoiadas em uma sequência de regras linguísticas, traduzidas por um conjunto de regras. As regras formuladas pelo ser humano são usualmente da forma:

*SE* (um conjunto de condições é satisfeito) *ENTÃO* (um conjunto de consequências pode ser inferido).

Para que seja possível formalizar as regras fuzzy, é necessário definir o conceito de relação fuzzy.

**Definição 2.8.** *Uma relação fuzzy  $R$  é um subconjunto fuzzy do produto cartesiano entre conjuntos universos  $X_1, X_2, \dots, X_n$ , ou seja,  $R \subseteq X_1 \times X_2 \times \dots \times X_n$ . O produto cartesiano fuzzy  $A_1 \times A_2 \times \dots \times A_n$  dos subconjuntos fuzzy  $A_1, A_2, \dots, A_n$  de  $X_1, X_2, \dots, X_n$ , respectivamente, é a relação fuzzy  $R$  cuja função de pertinência é dada por:*

$$\mu_R(x_1, x_2, \dots, x_n) = \min(\mu_{A_1}(x_1), \mu_{A_2}(x_2), \dots, \mu_{A_n}(x_n)).$$

### 2.2.1 Regras e Inferência Fuzzy

Uma regra fuzzy é uma sentença da forma “SE  $X$  é  $A$  ENTÃO  $Y$  é  $B$ ”, em que  $A$  e  $B$  são conjuntos fuzzy em  $X$  e  $Y$ , respectivamente. Tal regra pode ser interpretada como uma relação fuzzy  $R$  entre os conjuntos  $A$  e  $B$  cuja função de pertinência  $\mu_R(x, y)$  depende de  $\mu_A(x)$  e  $\mu_B(y)$  para cada  $(x, y) \in X \times Y$ . Em geral utiliza-se a função:

$$\mu_R(x, y) = \min(\mu_A(x), \mu_B(y)).$$

Na maioria dos casos, as regras fuzzy são formuladas em termos de variáveis linguísticas, ou seja, variáveis que assumem valores qualitativos representados por termos linguísticos, como por exemplo, “SE a área é grande ENTÃO a produção é alta”.

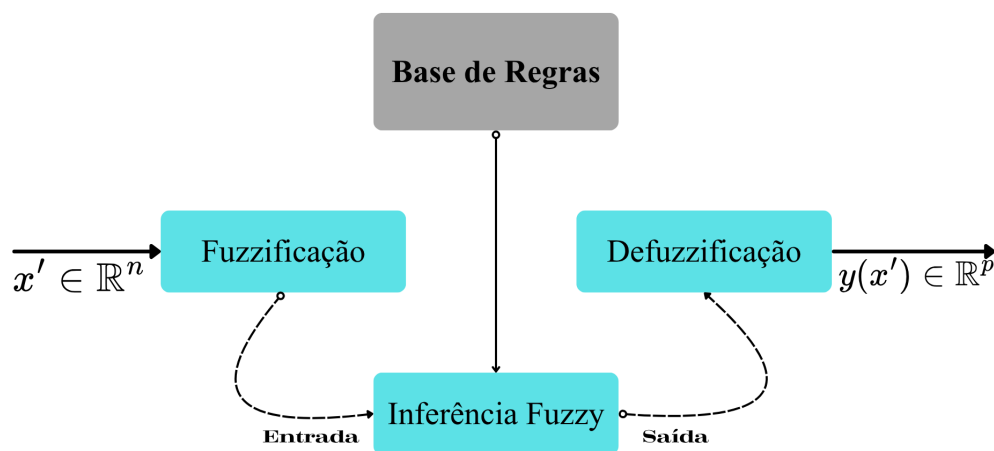
**Definição 2.9.** *Uma variável linguística é uma variável cujo o valor é expresso qualitativamente por termos linguísticos, que são representados por conjuntos fuzzy e quantitativamente por uma função de pertinência.*

Por exemplo, a variável linguística “área” pode assumir os termos linguísticos “pequena”, “média” e “grande” representados por conjuntos fuzzy com funções de pertinência que descrevem o grau de pertencimento de cada valor de área a esses termos.

### 2.2.2 Componentes de um SBRF

Um SBRF é composto por quatro componentes: a fuzzificação dos dados de entrada, uma coleção de regras, chamada base de regras, um método de inferência fuzzy e a defuzzificação, que fornece um número real como saída. Estes componentes são conectados como mostrado na Figura 2.5.

Figura 2.5: Arquitetura de um Sistema Baseado em Regras Fuzzy (SBRF).



Fonte: Adaptado de Jafelice, Barros e Bassanezi (2023).

O SBRF pode ser visto com um mapeamento entre a entrada e a saída da forma  $y = f(x)$ ,  $x \in \mathbb{R}^n$  e  $y \in \mathbb{R}^p$ . Os componentes do SBRF realizam as seguintes funções:

- A **fuzzificação** realiza a transformação dos valores de entrada em conjuntos fuzzy com seus respectivos domínios. A atuação de um especialista é necessária para definir as funções de pertinência que são utilizadas na fuzzificação, bem como os termos linguísticos que são associados a cada variável de entrada.
- A **base de regras** contém um conjunto de regras fuzzy que descrevem o comportamento do sistema, essas regras são formuladas por especialistas com base em seu conhecimento sobre o domínio do problema. A base de regras descreve relações entre as variáveis linguísticas para serem utilizadas pelo método de inferência fuzzy.
- O **método de inferência fuzzy** fornece a saída a partir de cada entrada fuzzy e da relação definida pela base de regras. O método de inferência fuzzy pode ser visto como um mecanismo de mapeamento entre os conjuntos fuzzy de entrada e os conjuntos fuzzy

de saída, utilizando as regras da base de regras para inferir a saída correspondente a cada entrada. Neste trabalho é utilizado o método de inferência de Takagi-Sugeno, que é um dos métodos mais comuns.

- **Método de Takagi-Sugeno:** Neste caso, o consequente de cada regra é uma função das variáveis de entrada. Por exemplo, pode-se supor que a função que mapeia a entrada e a saída para cada regra é uma combinação linear das entradas, ou seja:

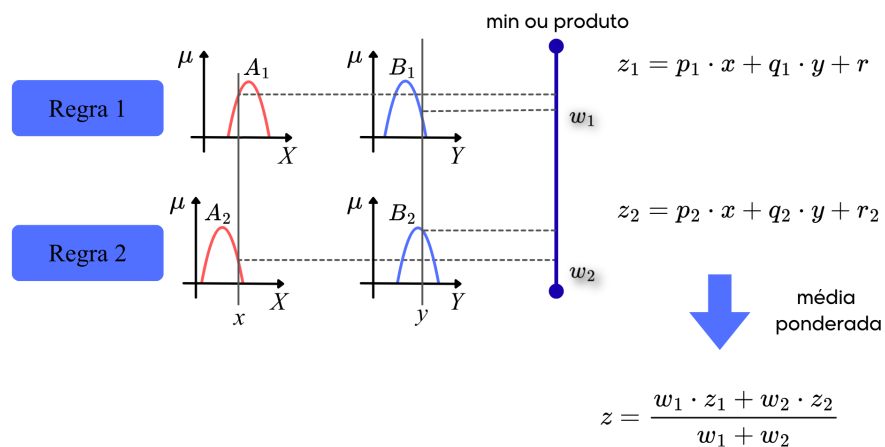
$$z = a_1x_1 + a_2x_2 + \cdots + a_nx_n + b.$$

Assim, as regras do método de Takagi-Sugeno podem ser formuladas da seguinte forma:

**Regra  $r$  :** SE  $x_1$  é  $A_1$  E  $x_2$  é  $A_2 \cdots x_n$  é  $A_n$  ENTAO  $z = f_r(x_1, x_2, \dots, x_n)$ .

Na Figura 2.6 é ilustrado um exemplo do método de Takagi-Sugeno, em que duas regras são aplicadas para inferir a saída  $z$  a partir de dois valores de entradas  $x$  e  $y$ .

Figura 2.6: Exemplo de inferência fuzzy utilizando o método de Takagi-Sugeno.



Fonte: Adaptado de Jafelice, Barros e Bassanezi (2023).

- A **defuzzificação** converte os conjuntos fuzzy de saída em números reais. Sistemas fuzzy, em geral, a saída é um conjunto fuzzy, assim, deve-se escolher um método de defuzzificação para obter um número real a partir do conjunto fuzzy de saída. Para o método de Takagi-Sugeno, o método de defuzzificação é dado pela média ponderada das saídas de cada regra, usando-se o grau de ativação destas regras como ponderação, ou seja:

$$z = \frac{\sum_{r=1}^R w_r f_r(x_1, x_2, \dots, x_n)}{\sum_{r=1}^R w_r},$$

em que  $R$  é o número de regras,  $w_r$  é o grau de ativação da regra  $r$  e  $f_r(x_1, x_2, \dots, x_n)$  é

a função de saída da regra  $r$ . É possível observar o cálculo na Figura 2.6 para obter “ $z$ ”.

**Exemplo 2.8.** *Considere um sistema de controle de velocidade de um ventilador com duas variáveis de entrada: temperatura  $x$  (em °C) e umidade do ar  $y$  (em %). O objetivo é determinar a velocidade  $z$  do ventilador (em Rotações por Minuto - RPM) utilizando um SBRF com o método de Takagi-Sugeno.*

As variáveis linguísticas são definidas pelas seguintes funções de pertinência trapezoidais:

- Temperatura “Baixa” ( $A_1$ ):

$$\mu_{A_1}(x) = \begin{cases} 1, & \text{se } x \leq 20 \\ \frac{25-x}{5}, & \text{se } 20 < x < 25 \\ 0, & \text{se } x \geq 25 \end{cases}.$$

- Temperatura “Média” ( $A_2$ ):

$$\mu_{A_2}(x) = \begin{cases} 0, & \text{se } x < 20 \text{ ou } x \geq 35 \\ \frac{x-20}{5}, & \text{se } 20 \leq x < 25 \\ 1, & \text{se } 25 \leq x < 30 \\ \frac{35-x}{5}, & \text{se } 30 \leq x < 35 \end{cases}.$$

- Temperatura “Alta” ( $A_3$ ):

$$\mu_{A_3}(x) = \begin{cases} 0, & \text{se } x < 30 \\ \frac{x-30}{5}, & \text{se } 30 \leq x < 35 \\ 1, & \text{se } x \geq 35 \end{cases}.$$

- Umidade “Baixa” ( $B_1$ ):

$$\mu_{B_1}(y) = \begin{cases} 1, & \text{se } y < 20 \\ \frac{40-y}{20}, & \text{se } 20 \leq y < 40 \\ 0, & \text{se } y \geq 40 \end{cases}.$$

- Umidade “Média” ( $B_2$ ):

$$\mu_{B_2}(y) = \begin{cases} 0, & \text{se } y < 20 \text{ ou } y \geq 80 \\ \frac{y-20}{20}, & \text{se } 20 \leq y < 40 \\ 1, & \text{se } 40 \leq y < 60 \\ \frac{80-y}{20}, & \text{se } 60 \leq y < 80 \end{cases}.$$

- Umidade “Alta” ( $B_3$ ):

$$\mu_{B_3}(y) = \begin{cases} 0, & \text{se } y < 60 \\ \frac{y-60}{20}, & \text{se } 60 \leq y < 80 \\ 1, & \text{se } y \geq 80 \end{cases}.$$

As regras fuzzy pelo método de Takagi-Sugeno são:

$$\left\{ \begin{array}{l} \textbf{Regra 1} : \text{SE } (x \text{ é } A_1 \text{ E } y \text{ é } B_1) \text{ ENTÃO } z_1 = 5x + 2y \\ \textbf{Regra 2} : \text{SE } (x \text{ é } A_2 \text{ E } y \text{ é } B_2) \text{ ENTÃO } z_2 = x + y \\ \textbf{Regra 3} : \text{SE } (x \text{ é } A_3 \text{ E } y \text{ é } B_3) \text{ ENTÃO } z_3 = 2x + 3y \\ \textbf{Regra 4} : \text{SE } (x \text{ é } A_1 \text{ E } y \text{ é } B_2) \text{ ENTÃO } z_4 = 3x + y \\ \textbf{Regra 5} : \text{SE } (x \text{ é } A_2 \text{ E } y \text{ é } B_1) \text{ ENTÃO } z_5 = 4x + 2y \\ \textbf{Regra 6} : \text{SE } (x \text{ é } A_3 \text{ E } y \text{ é } B_1) \text{ ENTÃO } z_6 = 2x + y \\ \textbf{Regra 7} : \text{SE } (x \text{ é } A_1 \text{ E } y \text{ é } B_3) \text{ ENTÃO } z_7 = 3x + 4y \\ \textbf{Regra 8} : \text{SE } (x \text{ é } A_2 \text{ E } y \text{ é } B_3) \text{ ENTÃO } z_8 = 4x + 3y \\ \textbf{Regra 9} : \text{SE } (x \text{ é } A_3 \text{ E } y \text{ é } B_2) \text{ ENTÃO } z_9 = 5x + y \end{array} \right. .$$

Para as entradas  $x = 35^\circ\text{C}$  e  $y = 70\%$ , os graus de pertinência são:

$$\mu_{A_1}(35) = 0, \quad \mu_{A_2}(35) = 0, \quad \mu_{A_3}(35) = 1.$$

$$\mu_{B_1}(70) = 0, \quad \mu_{B_2}(70) = 0,5, \quad \mu_{B_3}(70) = 0,5.$$

Os graus de ativação de cada regra, utilizando o operador do mínimo, são:

$$\left\{ \begin{array}{l} w_1 = \min(\mu_{A_1}(35), \mu_{B_1}(70)) = \min(0; 0) = 0 \\ w_2 = \min(\mu_{A_2}(35), \mu_{B_2}(70)) = \min(0; 0,5) = 0 \\ w_3 = \min(\mu_{A_3}(35), \mu_{B_3}(70)) = \min(1; 0,5) = 0,5 \\ w_4 = \min(\mu_{A_1}(35), \mu_{B_2}(70)) = \min(0; 0,5) = 0 \\ w_5 = \min(\mu_{A_2}(35), \mu_{B_1}(70)) = \min(0; 0) = 0 \\ w_6 = \min(\mu_{A_3}(35), \mu_{B_1}(70)) = \min(1; 0) = 0 \\ w_7 = \min(\mu_{A_1}(35), \mu_{B_3}(70)) = \min(0; 0,5) = 0 \\ w_8 = \min(\mu_{A_2}(35), \mu_{B_3}(70)) = \min(0; 0,5) = 0 \\ w_9 = \min(\mu_{A_3}(35), \mu_{B_2}(70)) = \min(1; 0,5) = 0,5 \end{array} \right. .$$

As saídas de cada regra são:

$$\left\{ \begin{array}{l} z_1 = 5 \cdot 35 + 2 \cdot 70 = 315 \\ z_2 = 35 + 70 = 105 \\ z_3 = 2 \cdot 35 + 3 \cdot 70 = 280 \\ z_4 = 3 \cdot 35 + 70 = 175 \\ z_5 = 4 \cdot 35 + 2 \cdot 70 = 280 \\ z_6 = 2 \cdot 35 + 70 = 140 \\ z_7 = 3 \cdot 35 + 4 \cdot 70 = 385 \\ z_8 = 4 \cdot 35 + 3 \cdot 70 = 350 \\ z_9 = 5 \cdot 35 + 70 = 245 \end{array} \right. .$$

A saída final é obtida pela média ponderada:

$$\begin{aligned} z &= \frac{w_1 \cdot z_1 + w_2 \cdot z_2 + w_3 \cdot z_3 + w_4 \cdot z_4 + w_5 \cdot z_5 + w_6 \cdot z_6 + w_7 \cdot z_7 + w_8 \cdot z_8 + w_9 \cdot z_9}{w_1 + w_2 + w_3 + w_4 + w_5 + w_6 + w_7 + w_8 + w_9} = \\ &= \frac{0 \cdot 315 + 0 \cdot 105 + 0,5 \cdot 280 + 0 \cdot 175 + 0 \cdot 280 + 0 \cdot 140 + 0 \cdot 385 + 0 \cdot 350 + 0,5 \cdot 245}{0 + 0 + 0,5 + 0 + 0 + 0 + 0 + 0 + 0,5} = \\ &= \frac{140 + 122,5}{1} = 262,5 \text{ RPM.} \end{aligned}$$

Portanto, para uma temperatura de 35°C e umidade do ar de 70%, o SBRF com o método de inferência Takagi-Sugeno determina que a possibilidade da velocidade do ventilador é de 262,5 RPM, uma boa velocidade para refrescar o ambiente.

## 2.3 ANFIS

O Sistema Adaptativo de Inferência Neuro-Fuzzy, do inglês *Adaptative Neuro-Fuzzy Inference System* (ANFIS), é uma técnica híbrida em inteligência artificial, que combina a capacidade de aprendizado das redes neurais artificiais com a representação de conhecimento e raciocínio aproximado dos sistemas baseados em regras fuzzy (JAFELICE; BARROS; BASSANEZI, 2023). Essa abordagem permite determinar um SBRF a partir de conhecimentos obtidos por meio de um processo de treinamento, utilizando um conjunto de dados de entrada e saída, sendo o principal método de inferência utilizado o método de Takagi-Sugeno (JANG, 1993). O ANFIS é composto por cinco camadas, cada uma com uma função específica.

**Exemplo 2.9.** Para exemplificar cada camada, considere o sistema com duas entradas,  $x_1$  e  $x_2$ , e uma saída  $y$ , com quatro regras fuzzy do tipo Takagi-Sugeno, ou seja:

**Regra 1:** SE  $x_1$  é  $A_1^{(1)}$  E  $x_2$  é  $A_1^{(2)}$  ENTÃO  $y = f_{1,1}(x_1, x_2) = k_{10} + k_{11}x_1 + k_{12}x_2$ ;

**Regra 2:** SE  $x_1$  é  $A_1^{(1)}$  E  $x_2$  é  $A_2^{(2)}$  ENTÃO  $y = f_{1,2}(x_1, x_2) = k_{20} + k_{21}x_1 + k_{22}x_2$ ;

**Regra 3:** SE  $x_1$  é  $A_2^{(1)}$  E  $x_2$  é  $A_1^{(2)}$  ENTÃO  $y = f_{2,1}(x_1, x_2) = k_{30} + k_{31}x_1 + k_{32}x_2$ ;

**Regra 4:** SE  $x_1$  é  $A_2^{(1)}$  E  $x_2$  é  $A_2^{(2)}$  ENTÃO  $y = f_{2,2}(x_1, x_2) = k_{40} + k_{41}x_1 + k_{42}x_2$ .

em que  $A_1^{(1)}, A_2^{(1)}$  são os conjuntos fuzzy do universo  $X_1$ , associados à variável de entrada  $x_1$  e  $A_1^{(2)}, A_2^{(2)}$  são os conjuntos fuzzy do universo  $X_2$ , associados à variável de entrada  $x_2$ ; e  $k_{i0}, k_{i1}, k_{i2}$  são os parâmetros de cada regra, para  $i = 1, 2, 3, 4$ . Para este exemplo, considera-se que as funções de pertinência dos conjuntos fuzzy  $A_j^{(1)}$  e  $A_j^{(2)}$  são gaussianas, definidas pela equação (2.2). Na Figura 2.7 é ilustrada a arquitetura do ANFIS para o exemplo apresentado.

Na camada 1 os valores de entrada são transformados em graus de pertinência dos conjuntos fuzzy, ou seja:

$$\begin{cases} y_i^{(1)} = \mu_{A_i^{(1)}}(x_1) \\ y_{i+2}^{(1)} = \mu_{A_i^{(2)}}(x_2) \end{cases},$$

para  $i = 1, 2$ .

Na camada 2, as regras fuzzy são criadas com base no operador lógico “E” definido por uma t-norma, ou seja:

$$\begin{cases} y_{1,1}^{(2)} = w_{1,1} = \mu_{A_1^{(1)}}(x_1) \cdot \mu_{A_1^{(2)}}(x_2) \\ y_{1,2}^{(2)} = w_{1,2} = \mu_{A_1^{(1)}}(x_1) \cdot \mu_{A_2^{(2)}}(x_2) \\ y_{2,1}^{(2)} = w_{2,1} = \mu_{A_2^{(1)}}(x_1) \cdot \mu_{A_1^{(2)}}(x_2) \\ y_{2,2}^{(2)} = w_{2,2} = \mu_{A_2^{(1)}}(x_1) \cdot \mu_{A_2^{(2)}}(x_2) \end{cases};$$

em que  $w_{h,j}$  é o peso de ativação da regra associada à combinação  $(A_h^{(1)}, A_j^{(2)})$ , para  $h, j = 1, 2$ . Na camada 3 os pesos de cada regra são normalizados, ou seja:

$$y_{h,j}^{(3)} = \bar{w}_{h,j} = \frac{w_{h,j}}{w_{1,1} + w_{1,2} + w_{2,1} + w_{2,2}};$$

para  $h, j = 1, 2$ . Na camada 4 a parte “ENTÃO” das regras fuzzy são criadas, ou seja:

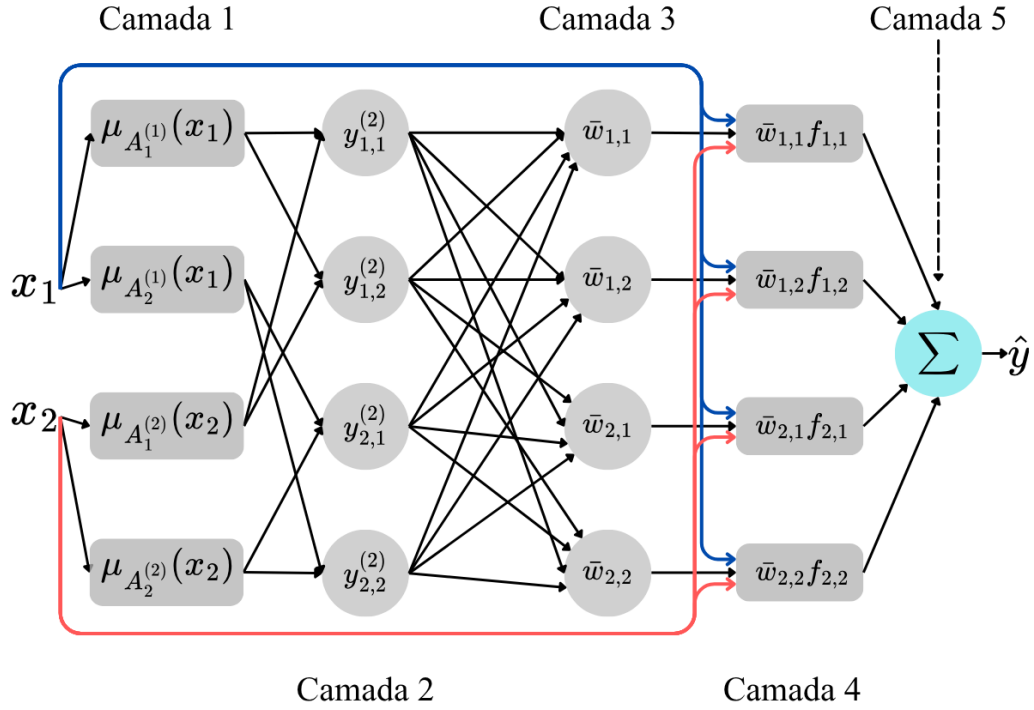
$$y_{h,j}^{(4)} = \bar{w}_{h,j} f_{h,j} = \bar{w}_{h,j} (k_{h0} + k_{h1}x_1 + k_{h2}x_2)$$

Para à camada 5, a saída é obtida pela média ponderada:

$$y = \frac{\sum_{h=1}^2 \sum_{j=1}^2 \bar{w}_{h,j} f_{h,j}}{\sum_{h=1}^2 \sum_{j=1}^2 \bar{w}_{h,j}}.$$

Observa-se que para dois antecedentes,  $x_1$  e  $x_2$ , e dois conjuntos fuzzy para cada antecedente,  $A_1^{(1)}, A_2^{(1)}$  e  $A_1^{(2)}, A_2^{(2)}$ , são criadas quatro regras fuzzy, ou seja,  $n^k = 2^2 = 4$  regras fuzzy. Para um número geral de  $k$  antecedentes e  $n$  conjuntos fuzzy para cada antecedente, o número de regras fuzzy criadas é dado por  $n^k$ .

Figura 2.7: Arquitetura do ANFIS



Fonte: Adaptado de Jafelice, Barros e Bassanezi (2023).

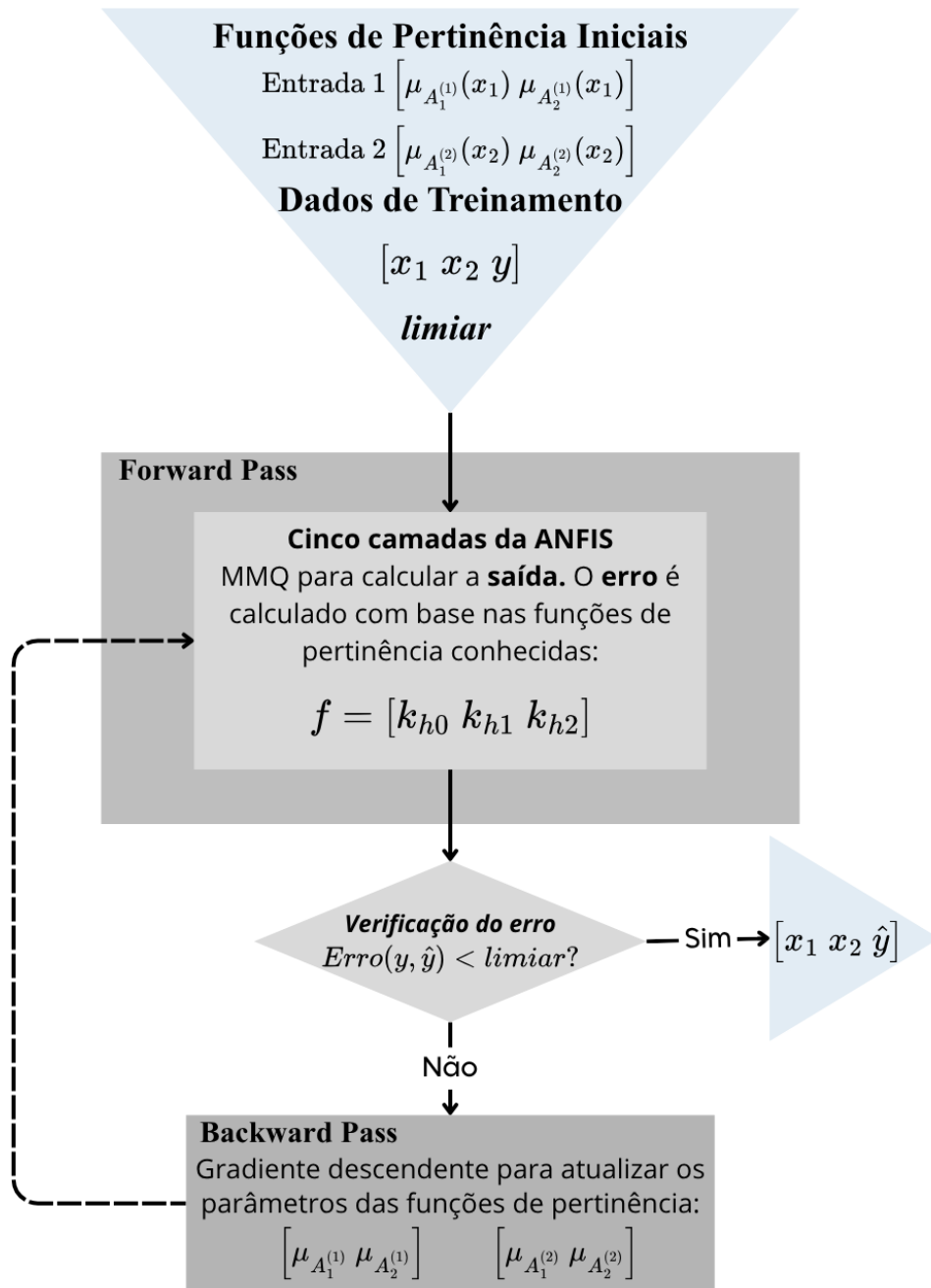
As setas na Figura 2.7 representam as conexões entre as camadas do ANFIS.

### 2.3.1 Treinamento

Assim, como em redes neurais MLPs, o ANFIS é treinado utilizando um conjunto de dados de entrada e saída, com o objetivo de ajustar os parâmetros das funções de pertinência e das regras fuzzy para minimizar o erro entre a saída prevista pelo modelo e a saída real. O treinamento é dividido em duas partes: *Forward Pass*, em português, Passe para Frente. Esta etapa é responsável por calcular a saída do modelo geralmente utilizando o Método dos Mínimos Quadrados (MMQ), para atualizar os parâmetros consequentes, ou seja,  $k_{h0}$ ,  $k_{h1}$ ,  $k_{h2}$ , para  $h = 1, 2, 3, 4$  (BURDEN, 2016). A etapa também é responsável por calcular o erro entre a saída prevista e a saída real.

A segunda etapa é o *Backward Pass*, em português, Passe para Trás. Esta etapa é responsável por ajustar os parâmetros das funções de pertinência, ou seja,  $\mu_{A_i^{(1)}}(x_1)$  e  $\mu_{A_i^{(2)}}(x_2)$ , para  $i = 1, 2$ , utilizando o método do gradiente descendente, ou suas variantes, para minimizar o erro calculado na etapa anterior. Na Figura 2.8 é apresentado o fluxo do treinamento do ANFIS, em que os dados de entrada e saída são utilizados para ajustar os parâmetros da rede, com o objetivo de minimizar o erro entre a saída prevista e a saída real. Na Figura 2.8 é apresentado o fluxo do treinamento do ANFIS.

Figura 2.8: Fluxo do treinamento do ANFIS.



Fonte: Adaptado de Jafelice, Barros e Bassanezi (2023).

O limiar presente na Figura 2.8 é utilizado como critério de parada do treinamento, ou seja, o treinamento é interrompido quando o erro entre a saída prevista e a saída real é menor que o limiar definido.

Neste capítulo foram apresentados os principais conceitos relacionados à teoria fuzzy, iniciando com a definição de conjuntos fuzzy e terminando com a definição da ANFIS. No próximo capítulo é introduzida a principal rede neural utilizada neste trabalho.

## Capítulo 3

# Estrutura das Redes Neurais U-Net e U-Net Fuzzy

Redes neurais convolucionais são frequentemente utilizadas em tarefas de classificação de imagens, nas quais o objetivo é atribuir uma classe à imagem como um todo. Entretanto, em muitas aplicações visuais, não basta apenas classificar a imagem inteira, é necessário também identificar com precisão a localização das estruturas de interesse. Nesses casos, utiliza-se a segmentação de imagens, isto é, o processo de classificar cada pixel da imagem em uma categoria específica. Para atender a essa necessidade, foi proposta a arquitetura U-Net (RONNEBERGER; FISCHER; BROX, 2015), uma rede neural convolucional desenvolvida especificamente para tarefas de segmentação. Sua estrutura é composta por duas partes principais: o *encoder*, responsável por extrair características da imagem de entrada, geralmente reduzindo suas dimensões espaciais e aumentando o número de canais, e o *decoder*, responsável por gerar a imagem a partir das características extraídas no *encoder*. Este processo permite a geração de uma máscara de segmentação, isto é, uma imagem que indica a presença ou ausência da região de interesse. Quando estas máscaras são desenvolvidas manualmente por especialistas, são conhecidas como *ground truth*. Essas máscaras são utilizadas como rótulos, ou seja, cada imagem enviada para treinamento da U-Net tem seu respectivo *ground truth*. Essa organização dá origem ao formato em “U”, que motivou o nome da arquitetura.

### 3.1 Noções Básicas da Arquitetura U-Net

Para compreender melhor a arquitetura da U-Net, é necessário entender as operações básicas envolvidas e como uma imagem é processada para gerar a máscara de segmentação. Para isso, o Exemplo 3.1 apresenta o funcionamento simplificado da U-Net, ilustrando as principais operações e etapas envolvidas no processo de segmentação de imagens. O exemplo a seguir é adaptado de TileStats (2025).

**Exemplo 3.1.** Considere uma imagem de entrada normalizada  $X$  de dimensões  $5 \times 5 \times 1$  e um *ground truth* de dimensões  $5 \times 5 \times 1$ . O objetivo deste exemplo é mostrar, de forma simplificada, como uma U-Net transforma a imagem de entrada  $X$  no *ground truth*. Para isso, considera-se

uma arquitetura reduzida, composta por um *encoder* com uma camada de convolução e uma camada de *Max pooling*, e por um *decoder* com uma camada de convolução transposta e uma camada de convolução final.

**Objetivo do exemplo.** O exemplo ilustra as principais etapas do funcionamento da U-Net, desde a extração de características no *encoder* até a criação da máscara de segmentação no *decoder*. Como se trata de uma versão simplificada, o foco está na compreensão do fluxo de processamento, e não na complexidade completa da arquitetura original.

**Simplificações adotadas.** Para facilitar a compreensão, são adotadas as seguintes simplificações: a imagem é monocromática, com dimensões  $5 \times 5 \times 1$ ; os pesos e os vieses já estão ajustados para o desenvolvimento da máscara de segmentação desejada. A rede possui apenas uma camada de convolução, uma camada de *Max pooling*, uma camada de convolução transposta e uma camada final de saída. As cores utilizadas nas imagens possuem apenas finalidade ilustrativa, servindo para destacar visualmente as operações realizadas. Elas não representam valores reais de pixel, uma vez que  $5 \times 5 \times 1$  corresponde a uma imagem monocromática.

Na Figura 3.1 são apresentadas a imagem de entrada  $X$  e *ground truth*. Neste exemplo, os pixels com valor “1” representam a região de interesse, enquanto os pixels com valor “0” representam o fundo. Valores intermediários, como “0,5”, representam áreas de transição ou incerteza.

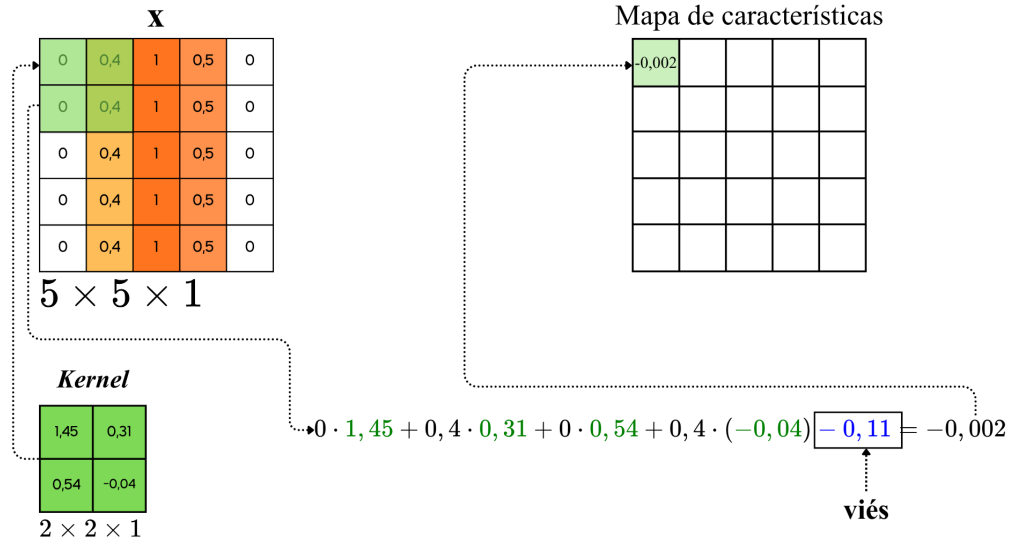
Figura 3.1: Imagem de entrada  $X$  e *ground truth*.

| <b>X</b>              |     |   |     |   | <b>Ground Truth</b>   |   |   |   |   |
|-----------------------|-----|---|-----|---|-----------------------|---|---|---|---|
| 0                     | 0,4 | 1 | 0,5 | 0 | 0                     | 1 | 1 | 1 | 0 |
| 0                     | 0,4 | 1 | 0,5 | 0 | 0                     | 1 | 1 | 1 | 0 |
| 0                     | 0,4 | 1 | 0,5 | 0 | 0                     | 1 | 1 | 1 | 0 |
| 0                     | 0,4 | 1 | 0,5 | 0 | 0                     | 1 | 1 | 1 | 0 |
| 0                     | 0,4 | 1 | 0,5 | 0 | 0                     | 1 | 1 | 1 | 0 |
| $5 \times 5 \times 1$ |     |   |     |   | $5 \times 5 \times 1$ |   |   |   |   |

Fonte: Elaborado pelo autor.

**Etapa 1: Convolução.** A primeira operação realizada no *encoder* é a convolução, equação (1.30). Nessa etapa, um *kernel* de dimensões  $2 \times 2 \times 1$  percorre a imagem de entrada  $X$ , combinando localmente os valores dos pixels, somando o viés e gerando um mapa de características, como mostrado na Figura 3.2.

Figura 3.2: Aplicação de um *kernel* de convolução na imagem de entrada  $X$ .

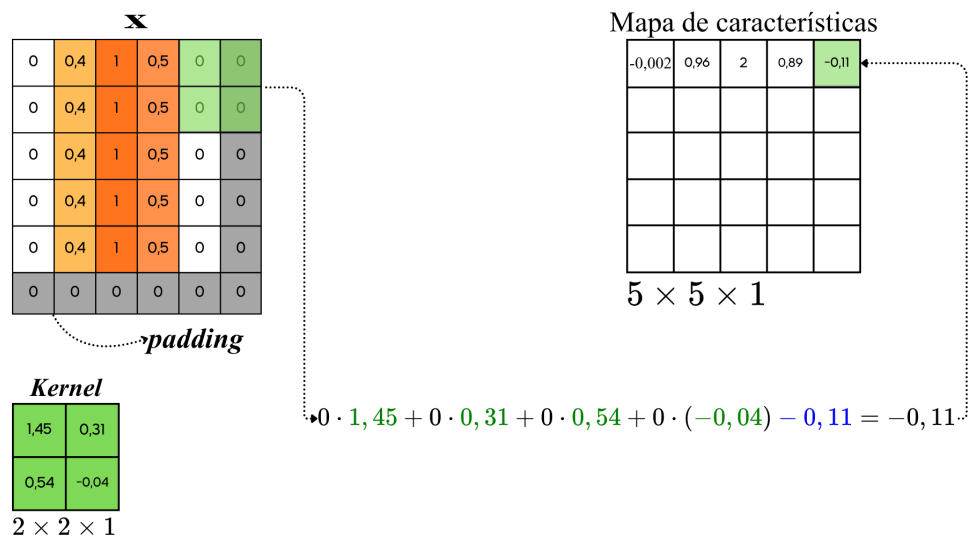


Fonte: Elaborado pelo autor.

Na Figura 3.2, os cálculos indicam os valores da imagem de entrada cobertos pelo *kernel*. A soma ponderada desses valores, acrescida do viés, gera cada elemento do mapa de características.

**Etapa 2: Padding.** Para que o mapa de características resultante mantenha as mesmas dimensões espaciais da imagem de entrada, isto é,  $5 \times 5 \times 1$ , aplica-se a técnica de *padding*, Subseção 1.3.6, adicionando uma borda de zeros ao redor da imagem. Neste exemplo, para simplificar a visualização, o preenchimento é adicionado apenas na parte inferior e à direita, como ilustrado em cinza na Figura 3.3.

Figura 3.3: Aplicação de *padding* na imagem de entrada  $X$ .

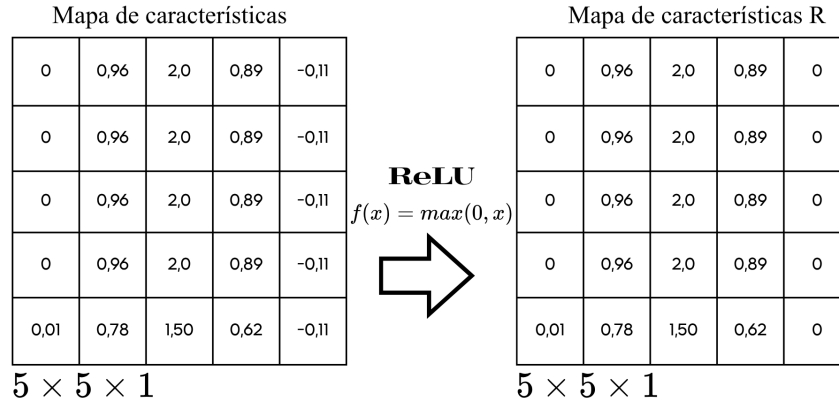


Fonte: Elaborado pelo autor.

**Etapa 3: Ativação ReLU.** Após a convolução, é aplicada uma função de ativação a cada valor

do mapa de características. Neste exemplo, utiliza-se a função ReLU, apresentada na equação (1.4). Sua função é introduzir não linearidade no modelo, permitindo que a rede represente padrões mais complexos. O resultado dessa operação é mostrado na Figura 3.4.

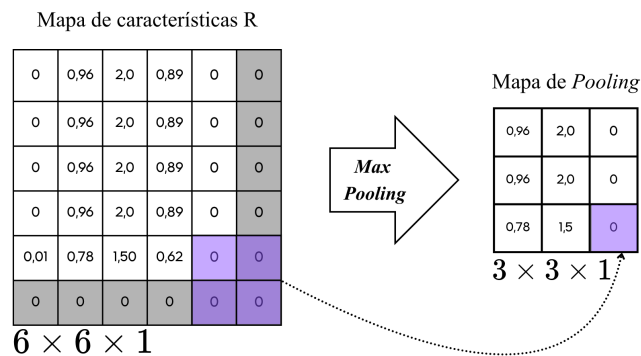
Figura 3.4: Aplicação da função de ativação ReLU no mapa de características.



Fonte: Elaborado pelo autor.

**Etapa 4: Max pooling.** Em seguida, o mapa de características é processado por uma camada de *Max pooling* com dimensões  $2 \times 2 \times 1$  e *stride* igual a 2, conforme descrito no Algoritmo 3. Essa operação reduz as dimensões espaciais do mapa, preservando os valores mais relevantes em cada região. Como a dimensão do mapa neste exemplo é ímpar, adiciona-se um *padding* para simplificar o processo, como mostrado na Figura 3.5.

Figura 3.5: Aplicação de uma camada de *Max pooling* no mapa de características resultante após a ReLU.

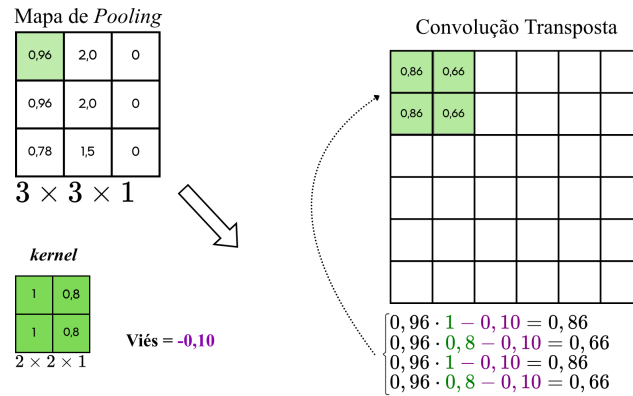


Fonte: Elaborado pelo autor.

Ao final dessa etapa, conclui-se a parte do *encoder*, resultando em um mapa de características com dimensões  $3 \times 3 \times 1$ . O processo de extração de características realizado pelo *encoder* é fundamental para que o *decoder* possa desenvolver a máscara de segmentação desejada.

**Etapa 5: Convolução transposta.** No *decoder*, o primeiro passo é expandir novamente as dimensões espaciais do mapa de características. Este procedimento é feito por meio de uma convolução transposta com *kernel* de dimensões  $2 \times 2 \times 1$ , conforme descrito no Algoritmo 4, como mostrado na Figura 3.6. Essa operação pode ser entendida como um processo de reconstrução espacial da informação comprimida no *encoder*. Cada valor do mapa de *pooling* é

Figura 3.6: Aplicação da convolução transposta no mapa de *pooling*.

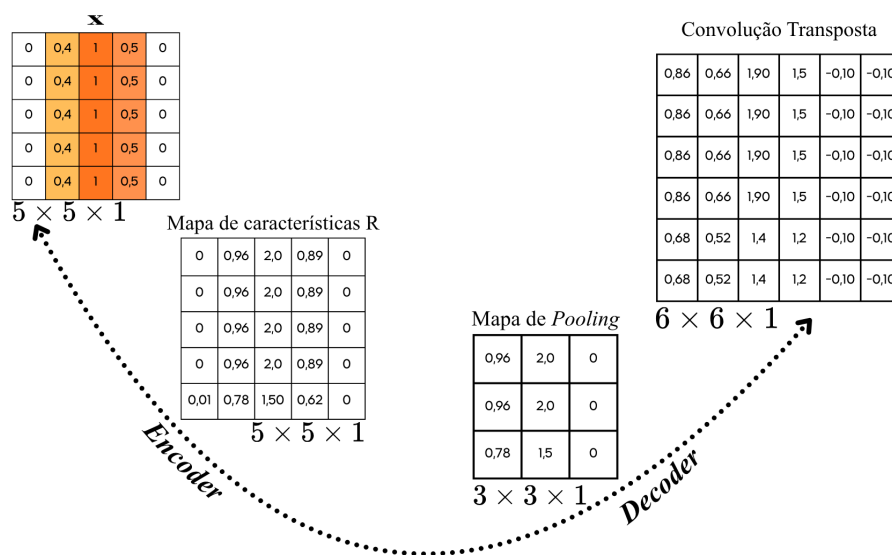


Fonte: Elaborado pelo autor.

multiplicado pelo *kernel* e adicionado às entradas do mapa de características que estão sendo cobertas pelo *kernel*, resultando em um mapa de características com dimensões  $6 \times 6 \times 1$ .

Na Figura 3.7 é resumida a dinâmica geral da U-Net: enquanto o *encoder* reduz progressivamente as dimensões espaciais para extrair características, o *decoder* expande essas dimensões para reconstruir a saída desejada.

Figura 3.7: Estrutura simplificada em forma de U da arquitetura U-Net.

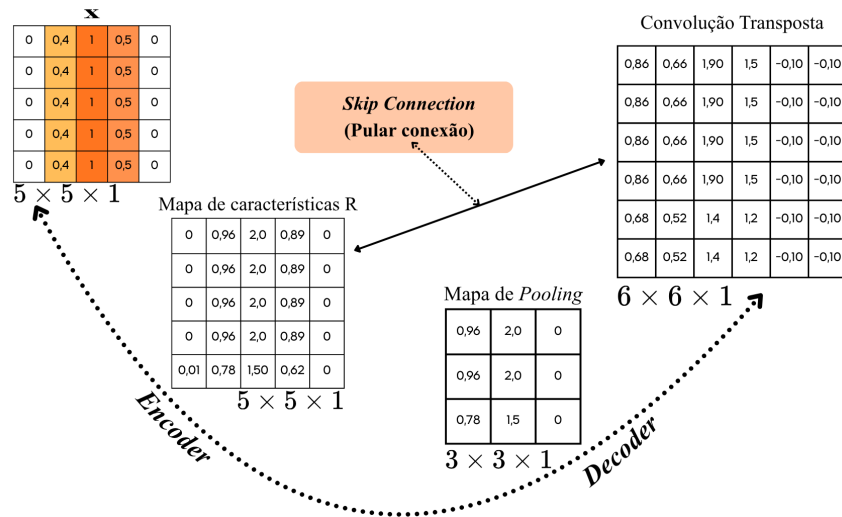


Fonte: Elaborado pelo autor.

**Etapa 6: Skip connection, copy and crop e concatenação.** Uma característica fun-

damental da U-Net é a presença das conexões de atalho (*skip connections*) entre camadas correspondentes do *encoder* e do *decoder*. Estas conexões permitem combinar as informações obtidas no *encoder*, como bordas e texturas, com as informações obtidas no *decoder*, favorecendo uma segmentação mais precisa. A ideia geral é ilustrada na Figura 3.8. As combinações são feitas por meio da operação de concatenação, que une os canais dos mapas de características extraídos pelo *encoder* com os canais dos mapas de características do *decoder* antes de cada etapa de convolução no *decoder*.

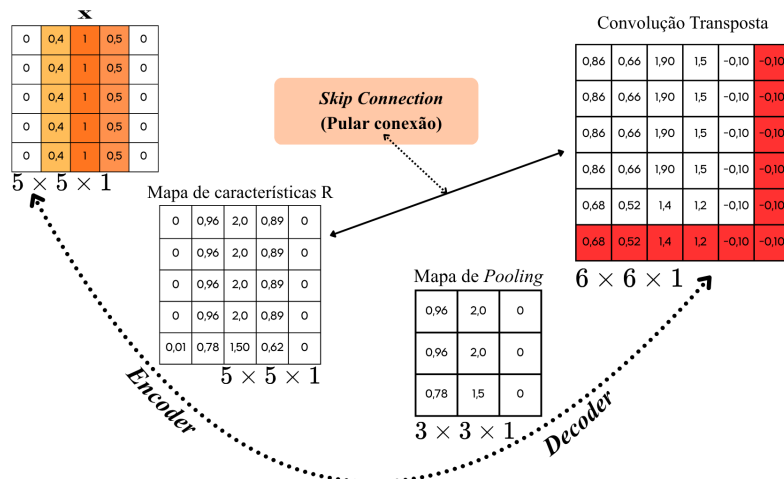
Figura 3.8: Exemplo de conexões de atalho (*skip connections*) na arquitetura U-Net.



Fonte: Elaborado pelo autor.

Para que os mapas de características possam ser concatenados, é necessário que possuam as mesmas dimensões espaciais. Quando isso não ocorre, aplica-se a técnica de Cópia e Corta, do inglês *copy and crop*, isto é, recorta-se o mapa de maior dimensão para compatibilizá-lo com o outro. Esse procedimento é ilustrado na Figura 3.9.

Figura 3.9: Exemplo da compatibilização de dimensões para a concatenação.



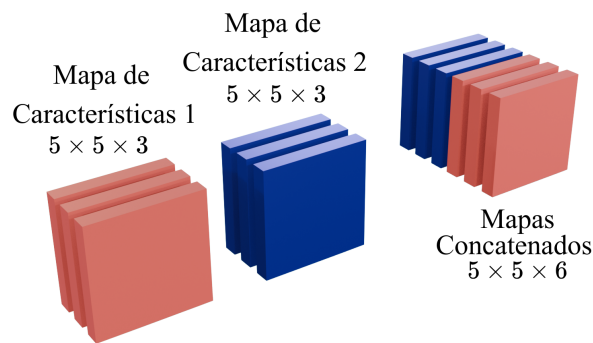
Fonte: Elaborado pelo autor.

A cor vermelha, presente na Figura 3.9, sinaliza quais linhas e colunas serão recortadas do mapa de características do *encoder*. Após a compatibilização das dimensões, a concatenação é realizada ao longo do eixo dos canais. Se dois mapas de características possuem dimensões espaciais iguais, a operação pode ser representada por:

$$X_1 = (A, L, C_1), \quad X_2 = (A, L, C_2) \quad \text{então} \quad X_1 \oplus X_2 = (A, L, C_1 + C_2),$$

em que  $A$  representa a altura,  $L$  a largura,  $C_1$  e  $C_2$  o número de canais, e  $\oplus$  a concatenação ao longo do eixo dos canais. Na Figura 3.10 é ilustrada essa operação.

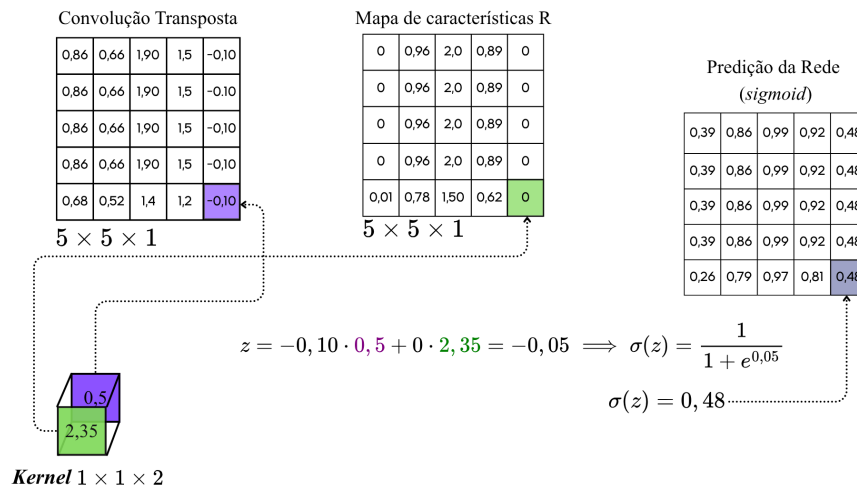
Figura 3.10: Exemplo de concatenação de dois mapas de características ao longo do eixo dos canais.



Fonte: Elaborado pelo autor.

**Etapa 7: Convolução final e limiar.** Após a concatenação, aplica-se uma última camada de convolução que não reduz a dimensão espacial, ou seja, o *kernel* é multicanais com dimensões de  $1 \times 1 \times 2$ , com dois canais, definição presente na Subseção 1.3.5.

Figura 3.11: Aplicação da última camada de convolução para gerar a saída da rede.

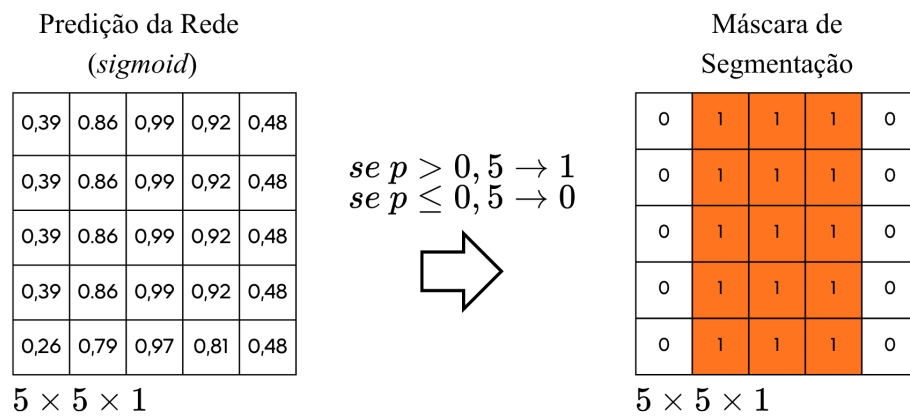


Fonte: Elaborado pelo autor.

Este procedimento é necessário pois o mapa de características após a concatenação possui dois canais, para este exemplo, como mostrado na Figura 3.11. Essa camada combina a informação dos canais e produz a saída da rede.

A aplicação da função *sigmoid* produz valores no intervalo  $[0, 1]$ , interpretados como probabilidades de cada pixel pertencer à região de interesse. Para que a saída da rede seja convertida na máscara de segmentação, aplica-se um limiar (*threshold*) para converter as probabilidades em valores binários, como mostrado na Figura 3.12.

Figura 3.12: Aplicação de um limiar (*threshold*) para gerar a máscara de segmentação.



Fonte: Elaborado pelo autor.

A máscara de segmentação resultante é então comparada com o *ground truth* utilizando métricas de similaridade, descritas na Seção 3.3, para avaliar a precisão da segmentação realizada pela U-Net.

O processo descrito neste exemplo é repetido para cada imagem de entrada durante o treinamento da rede, ajustando os pesos e vieses para minimizar a diferença entre a máscara de segmentação e o *ground truth*. Após o treinamento, a U-Net é capaz de generalizar e segmentar novas imagens de entrada, produzindo máscaras de segmentação precisas mesmo para imagens que não são vistas durante o treinamento.

**Resumo do exemplo.** Neste exemplo simplificado, a U-Net recebeu a imagem de entrada  $X$ , extraiu características relevantes no *encoder*, reduziu sua dimensão espacial por meio do *Max pooling*, reconstruiu a informação no *decoder* com auxílio das *skip connections* e, por fim, produziu a máscara de segmentação compatível com o *ground truth*. Embora simplificado, o exemplo evidencia os três elementos centrais da arquitetura: extração de características, reconstrução espacial e reaproveitamento de informações do *encoder* no *decoder*.

### 3.1.1 Arquitetura Original da U-Net

A rede convolucional desenvolvida por Ronneberger, Fischer e Brox (2015) é bem similar à arquitetura de rede utilizada no Exemplo 3.1, entretanto, a arquitetura completa da U-Net é composta por múltiplas camadas de convolução e *pooling* no *encoder*, e múltiplas camadas de convolução transposta no *decoder*. No total são quatro blocos de convolução e *pooling* no *encoder* e quatro blocos de convolução transposta no *decoder*, cada bloco contendo duas convoluções e um *pooling*, como é mostrado na Figura 3.13.

O número de *kernels* em cada camada de convolução aumenta à medida que se avança no *encoder*, começando com 64 *kernels* no primeiro bloco e dobrando a quantidade a cada bloco subsequente, chegando a 512 *kernels* no último bloco do *encoder*, os blocos do *encoder* são compostos por duas camadas de convolução seguidas de uma operação de *pooling*. No *decoder*, o número de *kernels* é reduzido pela metade a cada bloco subsequente, começando com 512 *kernels* no primeiro bloco do *decoder* e terminando com 64 *kernels* no último bloco, os blocos do *decoder* são compostos por uma camada de convolução transposta seguida de duas camadas de convolução. As conexões de atalho (*skip connections*) entre as camadas correspondentes do *encoder* e do *decoder* permitem que sejam feitas as concatenações dos mapas de características, e, caso a dimensão entre estes seja diferente, utiliza-se a técnica de *copy and crop*. No final do *encoder* tem-se um bloco conhecido como *bottleneck*, em português, gargalo. É o bloco mais profundo da rede, em que as características extraídas pela U-Net são processadas antes de serem enviadas para o *decoder*. O *bottleneck* é um bloco composto por duas camadas de convolução com o dobro da quantidade de *kernels* do último bloco do *encoder*, ou seja, 1024 *kernels*. A saída do *bottleneck* é então enviada para o *decoder* para gerar a máscara de segmentação.

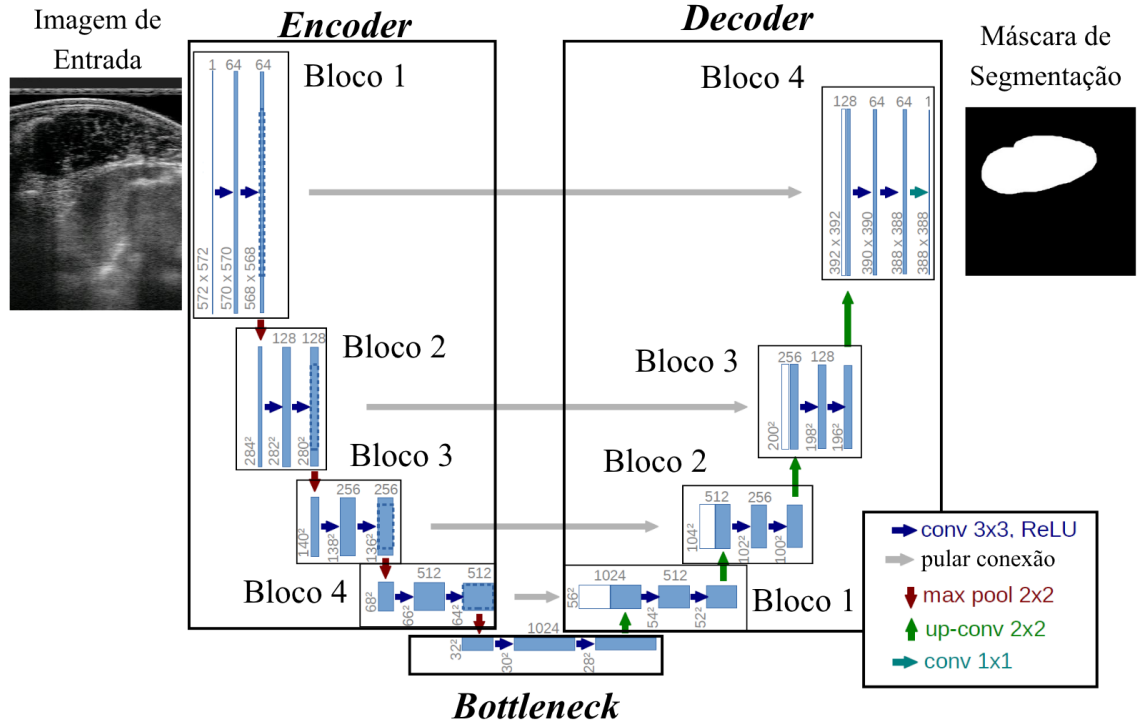
Na Figura 3.13 é apresentada a arquitetura completa da U-Net, cada bloco representa uma etapa de processamento, o número na parte superior do retângulo azul indica o número de canais presentes no mapa de características, enquanto o número na parte inferior do retângulo, indica as dimensões espaciais. As setas azuis representam a operação de convolução com *kernels* de dimensões  $3 \times 3 \times C_r^{(i)}$ , em que  $C_r^{(i)}$  é o número de canais do mapa de características resultante do bloco  $i$ , com função de ativação *ReLU*, as setas vermelhas representam a operação de *pooling* com *kernels* de dimensões  $2 \times 2 \times C_a^{(i)}$  e *stride* de 2, em que  $C_a^{(i)}$  é o número de canais do mapa de características do bloco  $i$ . As setas cinzas representam as conexões de atalho e as setas verdes representam as operações de convolução transposta com *kernels* de dimensões  $2 \times 2 \times C_d^{(i)}$ , em que  $C_d^{(i)} = 2 \cdot C_r^{(5-i)}$ , para  $i = 1, 2, 3, 4$ . A última camada de convolução utiliza um *kernel* de dimensões  $1 \times 1$  e função de ativação *sigmoid*, equação (1.3), para gerar a saída da rede, representado pela seta azul claro.

Como a saída é uma imagem em que cada pixel representa a probabilidade de pertencer ou não à região de interesse, um limiar, do inglês *threshold*, é aplicado para binarizar a saída da rede e produzir a máscara de segmentação, em que os pixels com valores de “1” representam a região de interesse e os pixels com valores de “0” representam o fundo.

Na U-Net original a técnica de *padding* utilizada é a “*valid*”, ou seja, não é adicionado nenhum preenchimento às bordas dos mapas de características, com isso em cada etapa de

convolução as dimensões dos mapas são reduzidas, conforme descrito na Subseção 1.3.6. Os cortes feitos para a concatenação, sinalizados por linhas azuis pontilhadas, nos mapas de características do *encoder* (lado direito da Figura 3.13) são feitos do centro para fora, ou seja, a parte central do mapa de características é mantida para a concatenação, sendo retiradas as mesmas quantidades de linhas e colunas de todos os lados. O total de parâmetros a serem otimizados, ou seja, pesos e vieses é de 27.548.993.

Figura 3.13: Arquitetura completa da U-Net.



Fonte: Adaptado de Ronneberger, Fischer e Brox (2015).

A imagem de entrada geralmente tem dimensões  $H_1 \times W_1 \times 1$ , com isso, para a primeira camada de convolução do *encoder*, tem-se 64 *kernels* com dimensões  $H_2 \times W_2 \times 1$ , gerando um mapa de características com dimensões  $H_3 \times W_3 \times 64$ . Para a segunda camada de convolução, tem-se 64 *kernels* com dimensões  $H_4 \times W_4 \times 64$ , gerando um mapa de características com dimensões  $H_5 \times W_5 \times 64$ , em que  $H_i$  e  $W_i$  representam a altura e a largura, respectivamente, para  $i = 1, 2, 3, 4, 5$ .

## 3.2 Modelos Propostos

Para melhorar a capacidade de segmentação da U-Net, são propostas duas arquiteturas derivadas do modelo original: a U-Net Multitarefa e a U-Net Fuzzy. Ambas compartilham uma mesma arquitetura base e diferem no tratamento aplicado ao *bottleneck* da rede.

### 3.2.1 Visão Geral das Modificações Propostas

As duas arquiteturas propostas preservam a estrutura geral da U-Net, composta por encoder, bottleneck e decoder. Em relação à arquitetura original, são adotadas modificações na quantidade de canais dos *kernels*, no tipo de *padding* e na forma de processamento das características extraídas no bottleneck. Na primeira proposta, a informação da raça do animal é incorporada por meio de um módulo multitarefa. Na segunda, além desta etapa, é incluído um módulo de inferência fuzzy para refinar as características antes da reconstrução da máscara de segmentação.

#### Modelo Base

A arquitetura base utilizada nos dois modelos é composta por quatro blocos no encoder e quatro blocos no decoder. No *encoder*, cada bloco é composto por duas camadas de convolução, a primeira camada de convolução tem  $k^{(i)}$  *kernels* de dimensões  $3 \times 3 \times C_i$ , em que  $k^{(i)}$  representa o número de *kernels* e  $C_i$  representa o número de canais do *kernel* no bloco  $i$ . Para a segunda camada de convolução, tem-se  $k^{(i)}$  *kernels* de dimensões  $3 \times 3 \times C_{i+1}$ , em que  $C_{i+1}$  representa o número de canais do *kernel* no bloco  $i + 1$ . Com  $k = (k^{(1)}, \dots, k^{(4)}) = (32, 64, 128, 128)$  e  $C = (C_1, \dots, C_5) = (1, 32, 64, 128, 128)$ . Cada camada de convolução possui *stride* de tamanho 1, *padding* “same” e função de ativação ReLU, seguidas por um *Max pooling* de dimensões  $2 \times 2 \times C_{i+1}$ , *stride* de tamanho 2, para reduzir a dimensão do mapa de características resultante, definições descritas nas Subseções 1.21, 1.3.7, 1.3.8, respectivamente. Optou-se pela convolução *same* pois melhores resultados são obtidos.

No *bottleneck*, tem-se duas camadas de convolução com *kernel* de dimensões  $3 \times 3 \times 256$ , *stride* de tamanho 1, *padding* “same” e função de ativação ReLU.

No *decoder*, a rede também é dividida em blocos, cada bloco é composto por uma camada de convolução transposta com  $k^{(i)}$  *kernels* de dimensões  $2 \times 2 \times C_i^{(0)}$ , *stride* de tamanho 2 e *padding* “same”, em que  $C_i^{(0)}$  representa o número de canais do *kernel* no bloco  $i$  para a convolução transposta, seguida por uma camada de concatenação para concatenar as características extraídas pelo *encoder* com as características extraídas pelo *decoder* e duas camadas de convolução com  $k^{(i)}$  *kernel* de dimensões  $3 \times 3 \times C_i^{(1)}$ , *stride* de tamanho 1, *padding* “same” e função de ativação ReLU, em que  $C_i^{(1)}$  representa o número de canais do *kernel* no bloco  $i$  para a convolução, com  $k = (k^{(1)}, \dots, k^{(4)}) = (128, 128, 64, 32)$ ,  $C^{(0)} = (C_1^{(0)}, \dots, C_4^{(0)}) = (256, 128, 128, 64)$  e  $C^{(1)} = (C_1^{(1)}, \dots, C_4^{(1)}) = (128, 128, 64, 32)$ . Como as dimensões dos mapas de características não são reduzidas em cada etapa de convolução do *encoder*, não é necessário realizar a técnica de *copy and crop* vista na U-Net original, Seção 3.1, pois as características extraídas pelo *encoder* e pelo *decoder* possuem as mesmas dimensões espaciais, como é mostrado na Figura 3.14.

A camada de saída é composta por uma camada de convolução com um único *kernel* de dimensões  $1 \times 1 \times 1$ , *stride* de tamanho 1 e função de ativação *sigmoid*, equação (1.3), para produzir a saída da rede, logo em seguida aplicando o *threshold* para binarizar a saída e gerar a máscara de segmentação.

## U-Net Multitarefa

Esta modificação envolve a adição de um novo bloco de classificação à arquitetura original da U-Net, permitindo que a rede aprenda a realizar tanto a segmentação quanto a classificação simultaneamente. O bloco de classificação é composta por unidades adicionais que processam as características extraídas pela U-Net para produzir uma saída de classificação no *bottleneck* (ZHU; ROHLING; SALCUDEAN, 2022). Um exemplo de aplicação é a segmentação de imagens de cancer de mama feitas por ultrassom, em que a U-Net é utilizada para segmentar a região de interesse (tumor) e a camada de classificação é utilizada para classificar o tipo do tumor (benigno (0) ou maligno (1)), a entrada da rede é (IMG, [0,1]), em que IMG representa a imagem de entrada e [0,1] representa os rótulos de classificação. A segunda entrada da rede geralmente está em codificação *one-hot*, que é uma técnica de codificação de rótulos categóricos em vetores binários, no caso do exemplo, benigno seria codificado para [1,0] e maligno para [0,1] (DATA SCIENCE ACADEMY, 2025).

O funcionamento do módulo pode-se ser entendido da seguinte forma: Após as duas operações de convolução do *bottleneck* a rede gera um mapa de característica de dimensões  $(B, H, W, C) = B \times H \times W \times C$ , em que  $B$  é o tamanho do lote (Subseção 1.2.2),  $H$  (altura) e  $W$  (largura) são as dimensões do mapa de características e  $C$  é o número de canais. A entrada da rede está no formato (IMG,  $[x_0, \dots, x_{n-1}]_{1 \times n}$ ), para  $x_k = 0$  e  $x_j = 1$ ,  $k \neq j$  e  $k, j = 0, \dots, n-1$ , na codificação *one-hot*. No módulo multitarefa é definida uma camada totalmente conectada, ou seja, uma rede neural com uma única camada oculta, com matriz de pesos  $W$  de entradas reais com dimensão  $C \times n$ . A saída da camada é então calculada pela seguinte operação matricial:

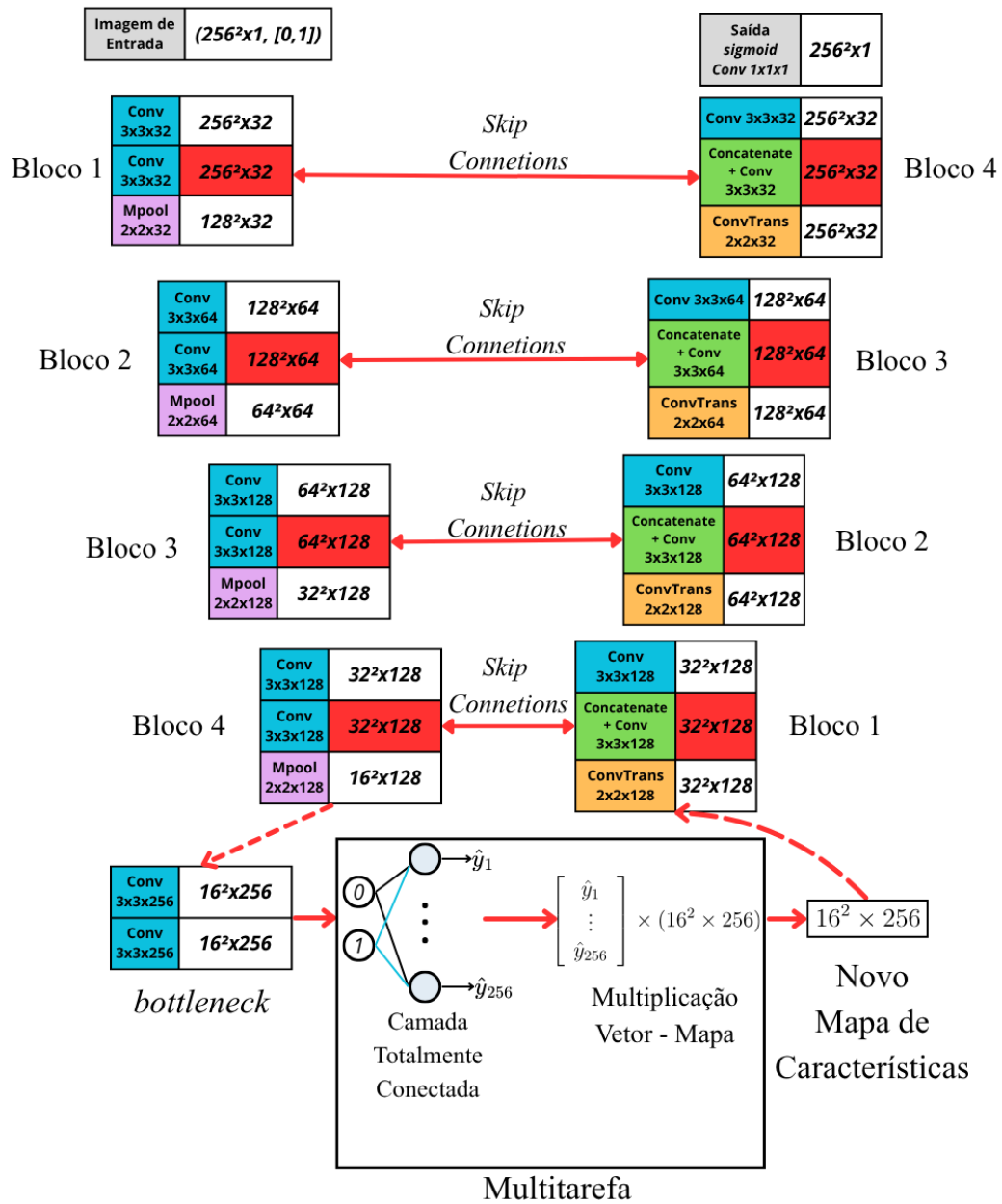
$$\tau(W_{C \times n} \odot x_{1 \times n}) = [\hat{y}_0, \hat{y}_1, \dots, \hat{y}_{C-1}]_{C \times 1},$$

em que “ $\odot$ ” representa a multiplicação matricial,  $x_{1 \times n} = [x_0, x_1, \dots, x_{n-1}]_{1 \times n}$  e  $\tau$  é a função de ativação utilizada. A saída da camada é um vetor coluna de tamanho  $C$ , em que cada elemento  $\hat{y}_i \in \mathbb{R}$ . Para finalizar o módulo multitarefa, o mapa de características original é multiplicado pelo vetor de saída da camada totalmente conectada. Após esse processo, o novo mapa de características é formado com a mesma dimensão do mapa de características de entrada e é enviada para o *decoder* da U-Net para obter a segmentação final ou, no caso deste trabalho, também é enviado para o módulo de inferência fuzzy, como é mostrado na Figura 3.15.

Na Figura 3.14 é mostrado a arquitetura da U-Net multitarefa utilizada e como as dimensões dos mapas de características vão se modificando ao longo da rede. A cor azul representa as camadas de convolução com *kernels* de dimensões  $3 \times 3 \times C_i$ , a cor roxa representa as camadas de *Max pooling*, a cor laranja representa as camadas de convolução transposta com *kernels* de dimensões  $3 \times 3 \times C_i$  e a cor verde representa as camadas de concatenação e convoluções com *kernels* de dimensões  $2 \times 2 \times C_{5-i}$ , para  $i = 1, \dots, 4$ . As setas indicam o caminho dos dados ao longo da rede, mostrando como as características extraídas pelo *encoder* são concatenadas com as características extraídas pelo *decoder* (*Skip Connection*) para produzir a máscara de segmentação final. O Multitarefa após o *bottleneck* é a representação visual das equações vistas

no módulo multitarefa.

Figura 3.14: Arquitetura da U-Net multitarefa utilizada e as dimensões dos mapas de características ao longo da rede.



Fonte: Elaborado pelo autor.

Na Figura 3.14 o mapa de características é representado como  $16^2 \times 256 = 16 \times 16 \times 256$ , sem o *lote* para facilitar a visualização. O módulo multitarefa está recebendo um mapa de características de dimensões  $16 \times 16 \times 256$  junto com o vetor  $[0,1]$ . A saída da camada totalmente conectada é um vetor de dimensões  $256 \times 1$  que é multiplicado pelo mapa de características original.

## U-Net Fuzzy

A U-Net Fuzzy compartilha a arquitetura base da U-Net Multitarefa, diferenciando-se pela inclusão de um módulo de inferência fuzzy (ANFIS) após o módulo multitarefa da rede. Este módulo é responsável por aplicar regras fuzzy para refinar as previsões da U-Net (VERMA *et al.*, 2026). A adição de um módulo fuzzy adiciona incerteza e flexibilidade ao processo de segmentação, permitindo que a rede aprenda a lidar com casos ambíguos ou ruidosos, como por exemplo um pixel que está próximo ao limite entre duas classes. Intuitivamente o módulo segue as etapas vistas na Seção 2.3, com algumas modificações para o contexto da U-Net. As etapas do módulo de inferência fuzzy são: pré-processamento, fuzzificação, inferência e defuzzificação.

Na etapa de pré-processamento, o mapa de características  $(B, H, W, C)$  recebido do módulo multitarefa é separado em vetores de canais de dimensões  $(B, 1, 1, C)$ . Por exemplo, caso o mapa de características tenha dimensões  $(1, 15, 15, 256)$ , então a quantidade de vetores gerados é  $15 \cdot 15 = 225$ , cada um com dimensões  $(1, 1, 1, 256)$ , ou seja,  $x_{1 \times 256}^{(i)} = [x_0, \dots, x_{255}]_{1 \times 256}$ , em que  $x_j$  é o valor de cada canal e  $x_{1 \times 256}^{(i)}$  são os vetores gerados, para  $i = 0, \dots, 224$  e  $j = 0, \dots, 255$ . Para que o módulo não introduza um número excessivo de parâmetros, é conveniente trabalhar com um número reduzido de canais, por isso, ao iniciar o módulo fuzzy, definem-se o número de antecedentes (número de entradas do módulo) e o número de conjuntos fuzzy para cada antecedente. Portanto, a etapa de pré-processamento é responsável por reduzir a quantidade de canais em cada vetor para o número de antecedentes definidos. O pré-processamento é dado por:

$$\bar{x}_{1 \times a}^{(i)} = x_{1 \times C}^{(i)} \odot W_{C \times a} + b_{1 \times a} = [\bar{x}_0, \bar{x}_1, \dots, \bar{x}_{a-1}]_{1 \times a},$$

em que  $W_{C \times a}$  é a matriz de pesos treináveis da camada de pré-processamento,  $x_{1 \times C}^{(i)}$  é o vetor de canais, para  $i = 0, \dots, (H \cdot W) - 1$ , “ $a$ ” é o número de antecedentes definidos para o módulo fuzzy,  $b_{1 \times a}$  é o vetor de vieses da camada de pré-processamento e  $\odot$  representa a multiplicação matricial. A matriz de pesos e o vetor de vieses são treináveis utilizando o método de *backpropagation*, Subseção 1.2.1. O resultado do pré-processamento é um vetor  $\bar{x}_{1 \times a}^{(i)}$ , que é enviado para a etapa de fuzzificação.

Na etapa de fuzzificação, cada valor do vetor  $\bar{x}_{1 \times a}^{(i)}$  receberá um grau de pertinência para cada conjunto fuzzy definido previamente, ou seja, as dimensões de  $\bar{x}_{1 \times a}^{(i)}$  são transformadas para  $\bar{x}_{1 \times a \times F}^{(i)}$ :

$$\bar{x}_{1 \times a \times F}^{(i)} = \begin{bmatrix} [\mu_0(\bar{x}_0^{(i)}), \dots, \mu_{F-1}(\bar{x}_0^{(i)})]_{1 \times F}, [\mu_0(\bar{x}_1^{(i)}), \dots, \mu_{F-1}(\bar{x}_1^{(i)})]_{1 \times F}, \\ \dots, [\mu_0(\bar{x}_{a-1}^{(i)}), \dots, \mu_{F-1}(\bar{x}_{a-1}^{(i)})]_{1 \times F} \end{bmatrix}_{1 \times a},$$

em que  $F$  é o número de conjuntos fuzzy definidos para cada antecedente e  $\mu_p(\bar{x}_j^{(i)})$  é o valor da função de pertinência para o  $p$ -ésimo conjunto fuzzy do  $j$ -ésimo antecedente, para  $p = 0, \dots, F - 1$  e  $j = 0, \dots, a - 1$ . As funções de pertinência são gaussianas, descritas na equação (2.2). Para cada antecedente  $\bar{x}_j^{(i)}$  e para cada conjunto fuzzy associado, a função de pertinência

possui um centro  $c_{j,\ell}$ , inicializado no intervalo  $[-1, 1]$ , e um desvio padrão  $\sigma_{j,\ell}$ , inicialmente igual a 0,5, com  $\ell = 0, \dots, F - 1$ . Todos esses parâmetros são treináveis utilizando o método de *backpropagation*, Subseção 1.2.1.

A etapa de inferência é dividida em duas partes: a parte “SE” e a parte “ENTÃO”. Para a primeira parte, a quantidade de regras fuzzy é definida como  $R = F^a$ , em que  $R$  é o número total de regras fuzzy. Observa-se que o número de regras cresce exponencialmente com o número de antecedentes, por isso é importante definir um número pequeno de antecedentes para o módulo fuzzy. Com isso, os valores do vetor  $\bar{x}_{1 \times a \times F}^{(i)}$  são combinados para produzir os pesos “ $w$ ”, também chamados de forças de ativação de cada regra. Por exemplo, considere  $a = 3$  antecedentes, dados por  $\bar{x}_0$ ,  $\bar{x}_1$  e  $\bar{x}_2$ , e  $F = 2$  conjuntos fuzzy para cada antecedente. Assim, para o primeiro antecedente têm-se os conjuntos  $A_1^{(0)}$  e  $A_2^{(0)}$ , para o segundo,  $A_1^{(1)}$  e  $A_2^{(1)}$ , e, para o terceiro,  $A_1^{(2)}$  e  $A_2^{(2)}$ . Nesse caso, o número total de regras é  $R = F^a = 2^3 = 8$ .

Suponha que, após a fuzzificação, sejam obtidos os seguintes graus de pertinência:

$$\begin{aligned}\mu_{A_1^{(0)}}(\bar{x}_0) &= 0,2, & \mu_{A_2^{(0)}}(\bar{x}_0) &= 0,6, \\ \mu_{A_1^{(1)}}(\bar{x}_1) &= 0,3, & \mu_{A_2^{(1)}}(\bar{x}_1) &= 0,7, \\ \mu_{A_1^{(2)}}(\bar{x}_2) &= 0,4, & \mu_{A_2^{(2)}}(\bar{x}_2) &= 0,5.\end{aligned}$$

Então,  $\bar{x}_{1 \times 3 \times 2} = \begin{bmatrix} [0, 2, 0, 6], [0, 3, 0, 7], [0, 4, 0, 5] \end{bmatrix}$ . As oito forças de ativação são então:

$$\begin{aligned}w_{1,1,1} &= \mu_{A_1^{(0)}}(\bar{x}_0)\mu_{A_1^{(1)}}(\bar{x}_1)\mu_{A_1^{(2)}}(\bar{x}_2) = 0,2 \cdot 0,3 \cdot 0,4 = 0,024, \\ w_{1,1,2} &= \mu_{A_1^{(0)}}(\bar{x}_0)\mu_{A_1^{(1)}}(\bar{x}_1)\mu_{A_2^{(2)}}(\bar{x}_2) = 0,2 \cdot 0,3 \cdot 0,5 = 0,030, \\ w_{1,2,1} &= \mu_{A_1^{(0)}}(\bar{x}_0)\mu_{A_2^{(1)}}(\bar{x}_1)\mu_{A_1^{(2)}}(\bar{x}_2) = 0,2 \cdot 0,7 \cdot 0,4 = 0,056, \\ w_{1,2,2} &= \mu_{A_1^{(0)}}(\bar{x}_0)\mu_{A_2^{(1)}}(\bar{x}_1)\mu_{A_2^{(2)}}(\bar{x}_2) = 0,2 \cdot 0,7 \cdot 0,5 = 0,070, \\ w_{2,1,1} &= \mu_{A_2^{(0)}}(\bar{x}_0)\mu_{A_1^{(1)}}(\bar{x}_1)\mu_{A_1^{(2)}}(\bar{x}_2) = 0,6 \cdot 0,3 \cdot 0,4 = 0,072, \\ w_{2,1,2} &= \mu_{A_2^{(0)}}(\bar{x}_0)\mu_{A_1^{(1)}}(\bar{x}_1)\mu_{A_2^{(2)}}(\bar{x}_2) = 0,6 \cdot 0,3 \cdot 0,5 = 0,090, \\ w_{2,2,1} &= \mu_{A_2^{(0)}}(\bar{x}_0)\mu_{A_2^{(1)}}(\bar{x}_1)\mu_{A_1^{(2)}}(\bar{x}_2) = 0,6 \cdot 0,7 \cdot 0,4 = 0,168, \\ w_{2,2,2} &= \mu_{A_2^{(0)}}(\bar{x}_0)\mu_{A_2^{(1)}}(\bar{x}_1)\mu_{A_2^{(2)}}(\bar{x}_2) = 0,6 \cdot 0,7 \cdot 0,5 = 0,210.\end{aligned}$$

De forma geral, cada regra fuzzy é associada a uma combinação  $(n_0, \dots, n_{a-1})$ , com  $n_j \in \{1, \dots, F\}$  indicando qual conjunto fuzzy é selecionado para o  $j$ -ésimo antecedente. Assim, a força de ativação da regra associada a essa combinação é dada por:

$$w_{n_0, \dots, n_{a-1}} = \prod_{j=0}^{a-1} \mu_{A_{n_j}^{(j)}}(\bar{x}_j^{(i)}).$$

em que  $A_{n_j}^{(j)}$  é o  $n_j$ -ésimo conjunto fuzzy do  $j$ -ésimo antecedente. Em outras palavras, a força de ativação é obtida pela multiplicação dos graus de pertinência correspondentes a uma combinação

específica de conjuntos fuzzy. Para resumir a notação, defini-se  $w_r$  como sendo a força de ativação associado à regra  $r$ . Em seguida, as forças de ativação são normalizadas utilizando o seguinte cálculo:

$$\bar{w}_r = \frac{w_r}{\sum_{k=1}^R w_k + \epsilon},$$

em que  $\bar{w}_r$  é a força de ativação normalizada da regra  $r$  e  $\epsilon$  é um valor pequeno adicionado para evitar a divisão por zero.

A parte “ENTÃO” é definida por uma função linear das entradas, baseada no método de Takagi-Sugeno, ou seja:

$$\begin{cases} (f^{(1)}(x_{1 \times C}^{(i)}))_{1 \times C} = x_{1 \times C}^{(i)} * K_{1 \times C}^{(1)} + (b^{(1)})_{1 \times C}, \\ \vdots \\ (f^{(R)}(x_{1 \times C}^{(i)}))_{1 \times C} = x_{1 \times C}^{(i)} * K_{1 \times C}^{(R)} + (b^{(R)})_{1 \times C}, \end{cases}$$

em que  $K^{(r)}$  é a matriz de pesos treináveis da regra  $r$ ,  $b^{(r)}$  é o vetor de vieses treináveis da regra  $r$ ,  $x_{1 \times C}^{(i)}$  é o  $i$ -ésimo vetor de entrada do módulo de inferência fuzzy e o operador “\*” simboliza a multiplicação elemento a elemento entre os vetores. Nesta etapa, o número de parâmetros treináveis é de  $R \times (C + C)$ , ou seja, para cada regra são treinados uma matriz de pesos de dimensões  $1 \times C$  e um vetor de vieses de dimensões  $1 \times C$ . Na etapa de defuzzificação, para gerar a saída da inferência, cada regra é multiplicada pela força de ativação normalizada da regra:

$$y_r^{(i)}_{1 \times C} = \bar{w}_r \cdot f^{(r)}(x_{1 \times C}^{(i)})_{1 \times C},$$

em que  $y_r^{(i)}$  é a saída para a  $r$ -ésima regra do módulo de inferência fuzzy do  $i$ -ésimo vetor de entrada e  $\bar{w}_r$  é a força de ativação normalizada da regra  $r$ . É por esse motivo que  $w_r$  é chamado de força de ativação, se  $w_r$  for baixo, então a normalização  $\bar{w}_r$  também tende a ser baixa, reduzindo a contribuição de  $f^{(r)}(x_{1 \times C}^{(i)})$  na saída final, mesmo que essa função produza um valor alto. O vetor de saída é então calculado pela soma ponderada das saídas de todas as regras, ou seja:

$$y_{1 \times C}^{(i)} = \sum_{r=1}^R y_r^{(i)},$$

A saída do módulo de inferência fuzzy para cada vetor processado é o vetor  $y_{1 \times C}^{(i)}$ , e após todos os vetores serem calculados, são reorganizados no formato  $(B, H, W, C)$  para formar o mapa de características de saída do módulo fuzzy.

No início do treinamento, o mapa de características de saída do módulo fuzzy é somente ruído, para evitar que esse ruído extra atrapalhe o processo de treinamento da U-Net, a saída do módulo fuzzy é somada ao mapa de características original da camada multitarefa para obter um mapa de características final:

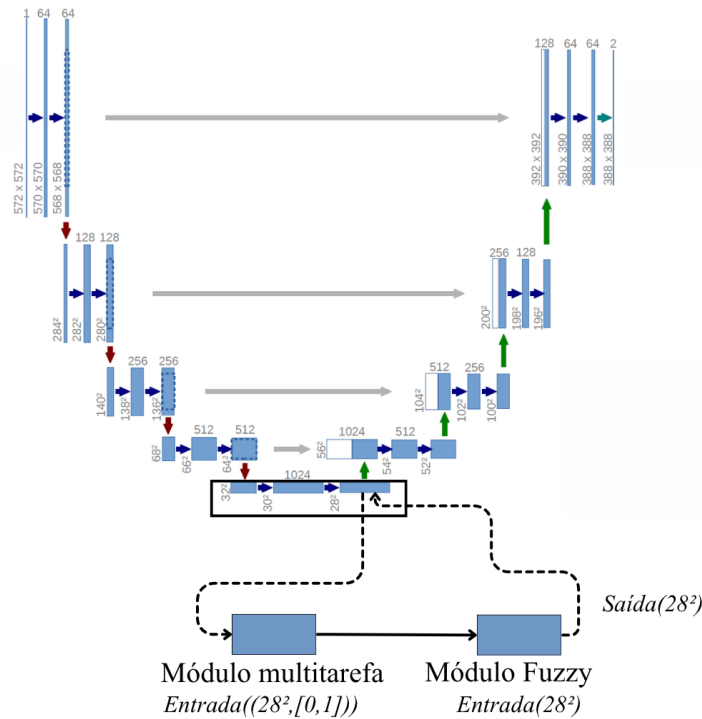
$$Y = Y_m + \alpha Y_f,$$

em que  $Y$  é o mapa de características final,  $Y_m$  é o mapa de características original da camada

multitarefa,  $Y_f$  é o mapa de características de saída do módulo fuzzy e  $\alpha$  é um hiperparâmetro que controla a influência do módulo fuzzy no mapa de características final. O valor de  $\alpha$  é definido como um valor pequeno, por exemplo, 0, 1, para que a influência do módulo fuzzy seja limitada no início do treinamento, permitindo que a U-Net aprenda a segmentar as imagens sem ser muito afetada pelo ruído gerado pelo módulo fuzzy. Conforme o treinamento avança, o valor de  $\alpha$  pode ser aumentado gradualmente para permitir que o módulo fuzzy tenha mais influência no processo de segmentação, refinando as previsões da U-Net à medida que o modelo aprende a lidar com casos ambíguos ou ruidosos.

Após essa etapa o mapa de características é processado pelo *decoder* da U-Net para obter a máscara de segmentação. Na Figura 3.15 é ilustrada a arquitetura da U-Net com a adição de uma camada Multitarefa e uma camada de inferência fuzzy.

Figura 3.15: Arquitetura da U-Net com a adição de uma camada Multitarefa e uma camada de inferência fuzzy.



Fonte: Adaptado de Ronneberger, Fischer e Brox (2015).

As setas mostram o fluxo do *bottleneck* pelas camadas adicionais, em que “28<sup>2</sup>” representa as dimensões do mapa de características.

### 3.3 Critérios de Avaliação

#### 3.3.1 Coeficiente de Dice

Quando a região de interesse em uma imagem é pequena em comparação ao plano de fundo, a métrica da acurácia pode ser enganosa, pois a maioria dos pixels da imagem pertencem ao

plano de fundo, o que pode levar a um resultado de alta acurácia mesmo que o modelo não esteja segmentando corretamente, ou seja, classificando cada pixel como pertencente ou não à região de interesse. Para lidar com esse problema, é comum utilizar métricas específicas para avaliar o resultado, como o coeficiente de Dice, que é uma medida de similaridade entre a máscara de segmentação predita e o *ground truth* (DICE, 1945). O coeficiente de Dice é calculado utilizando a seguinte fórmula:

$$Dice = \frac{2|A \cap B|}{|A| + |B|}, \quad (3.1)$$

em que  $A$  é o conjunto de pixels da máscara de segmentação predita e  $B$  é o conjunto de pixels do *ground truth*. O coeficiente de Dice varia entre 0 e 1, em que 1 indica sobreposição perfeita entre a máscara de segmentação e o *ground truth*, enquanto 0 indica que não há sobreposição entre a máscara de segmentação e o *ground truth*.

**Exemplo 3.2.** *Seja  $A = \{1, 2, 3, 4, 5, 6\}$  e  $B = \{1, 2, 3, 4, 9\}$ . A interseção entre os conjuntos  $A$  e  $B$  é  $A \cap B = \{1, 2, 3, 4\}$ , o que significa que existem 4 elementos em comum entre os dois conjuntos. O número total de elementos em  $A$  é  $|A| = 6$  e o número total de elementos em  $B$  é  $|B| = 5$ . Substituindo esses valores na fórmula do coeficiente de Dice, tem-se:*

$$Dice = \frac{2|A \cap B|}{|A| + |B|} = \frac{2 \cdot 4}{6 + 5} = \frac{8}{11} \approx 0,727.$$

*Portanto, o coeficiente de Dice para os conjuntos  $A$  e  $B$  é aproximadamente 0,727, indicando uma boa sobreposição entre os dois conjuntos, mas não uma correspondência perfeita.*

A métrica Dice é bastante utilizada em conjunto com a arquitetura U-Net para avaliar a qualidade da máscara de segmentação gerada pela rede. O objetivo é maximizar o valor do coeficiente de Dice, ou seja, aumentar a sobreposição entre a máscara de segmentação e o *ground truth* (MELO *et al.*, 2022).

### 3.3.2 Acurácia

A acurácia é uma métrica de avaliação utilizada para medir a proporção de pixels classificados corretamente pelo modelo em relação ao total de pixels na imagem. Para o contexto de segmentação de imagens, se  $C_T$  for os pixels da região de AOL traçada pelo especialista e  $C$  o conjunto dos pixels obtidos pelo método de predição. Os pixels definidos como verdadeiros positivos são  $VP = C_T \cap C$ , enquanto os pixels definidos como verdadeiros negativos são  $VN = \{p|p \notin C_T \text{ e } p \notin C\}$ . Os pixels falsos positivos são  $FP = \{p|p \notin C_T \text{ e } p \in C\}$  enquanto os pixels falsos negativos satisfazem  $FN = \{p|p \in C_T \text{ e } p \notin C\}$ . Com isso, a acurácia é calculada utilizando a fórmula (3.2),

$$Acurácia = \frac{VP + VN}{VP + VN + FP + FN}. \quad (3.2)$$

Para o contexto de segmentação de imagens, a acurácia nem sempre é tão precisa quanto o *Dice*. Caso a imagem de segmentação seja uma região pequena em comparação com o fundo da imagem, o modelo pode classificar a maioria dos pixels como fundo e ainda assim obter uma acurácia alta, mesmo que o coeficiente de Dice seja próximo de zero (MELO *et al.*, 2022).

### 3.3.3 Erro Médio Absoluto

O Erro Absoluto Médio, do inglês *Mean Absolute Error* (MAE), é uma métrica de avaliação utilizada para medir a média dos erros absolutos entre as medidas previstas pelo modelo e as medidas reais (AOL, EGL e EGG) (MELO *et al.*, 2022). Este é calculado utilizando a seguinte fórmula:

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i| \quad (3.3)$$

em que  $y_i$  são as medidas reais e  $\hat{y}_i$  são as medidas previstas pelo modelo e  $n$  é o número de amostras.

Com isso, neste capítulo foi apresentado os modelos propostos, as métricas de avaliação utilizadas e a metodologia empregada para a análise dos resultados. No próximo capítulo, são apresentados os resultados obtidos a partir da aplicação dos modelos propostos para o problema de segmentação de imagens de ultrassom bovino e uma análise detalhada dos mesmos.

## Capítulo 4

# Aplicação da U-Net e U-Net Fuzzy em Ultrassonografia Bovina

O processo para desenvolver uma rede neural envolve vários desafios, desde a obtenção de um conjunto de dados adequado, passando pelo pré-processamento das imagens, configuração dos modelos e avaliação dos resultados. Neste capítulo, são descritos os procedimentos adotados para cada uma dessas etapas, bem como os resultados obtidos e as análises realizadas.

### 4.1 Base de Dados e Planejamento Experimental

Para que os modelos propostos possam ser avaliados, é necessário um conjunto de dados que permita a comparação entre as arquiteturas. Esta seção apresenta o banco de dados utilizado, o processo de pré-processamento das imagens, a configuração dos modelos comparados e as métricas adotadas para a avaliação do desempenho.

#### 4.1.1 Contexto do Problema e Medidas Avaliadas

A raça Nelore é originária da Índia, onde seus ancestrais, os bovinos da raças Ongole, eram criados há milênios por sua rusticidade, resistência a climas tropicais e capacidade de adaptação a ambientes variados. Esses animais foram trazidos ao Brasil no final do século XIX, em um momento de expansão da pecuária nacional (ZANCANER, 2026). A raça Caracu, por sua vez, tem origem na região da Península Ibérica, chegando no Brasil logo após o descobrimento, os animais da raça Caracu são extremamente funcionais, com alta adaptabilidade e rusticidade, além de serem resistentes a doenças e parasitas (ABCCARACU, 2022).

Uma das principais formas de avaliar o desenvolvimento e a qualidade fenotípica dos animais é por meio da análise de imagens de ultrassonografia em que é possível obter informações da carcaça do animal. Dependendo da região do corpo do animal onde a imagem de ultrassom é obtida, é possível extrair diferentes informações, neste trabalho são utilizadas imagens de ultrassom da região do músculo *Longuissimus Dorsi* (Contra-filé), que localiza-se entre a 12<sup>a</sup> e 13<sup>a</sup> costela e da Garupa (WILSON, 1998). Na Figura 4.1 é mostrada a localização dessas



Na Figura 4.2(a) o especialista realiza a captura da imagem de ultrassom da região de olho de lombo. Na Figura 4.2(b) é apresentada a imagem obtida dessa região. Na Figura 4.2(c) o especialista realiza a captura da imagem da garupa. Na Figura 4.2(d) é apresentada a imagem de ultrassom da garupa.

### 4.1.2 Conjunto de Dados

O total de imagens disponibilizadas pelo Centro Avançado de Pesquisa e Desenvolvimento de Bovinos de Corte com a delimitação do contorno da região feita por uma especialista são 134 imagens de ultrassom (134 imagens da região de olho do lombo e 134 imagens da região da garupa), sendo 108 da raça Nelore e 27 da raça Caracu. Além das imagens disponibilizadas, foram enviadas também as medidas numéricas correspondentes à delimitação da AOL, EGL e EGG. As medidas estavam em um arquivo do Excel, em que a primeira coluna continha o número de identificação do animal, a segunda coluna continha o indicador da AOL, a terceira coluna continha o indicador da EGL e a quarta coluna continha o indicador da EGG. Essas medidas são utilizadas para calcular o MAE entre as medidas preditas pelo modelo e as medidas reais.

### 4.1.3 Pré-processamento e Máscaras de Segmentação

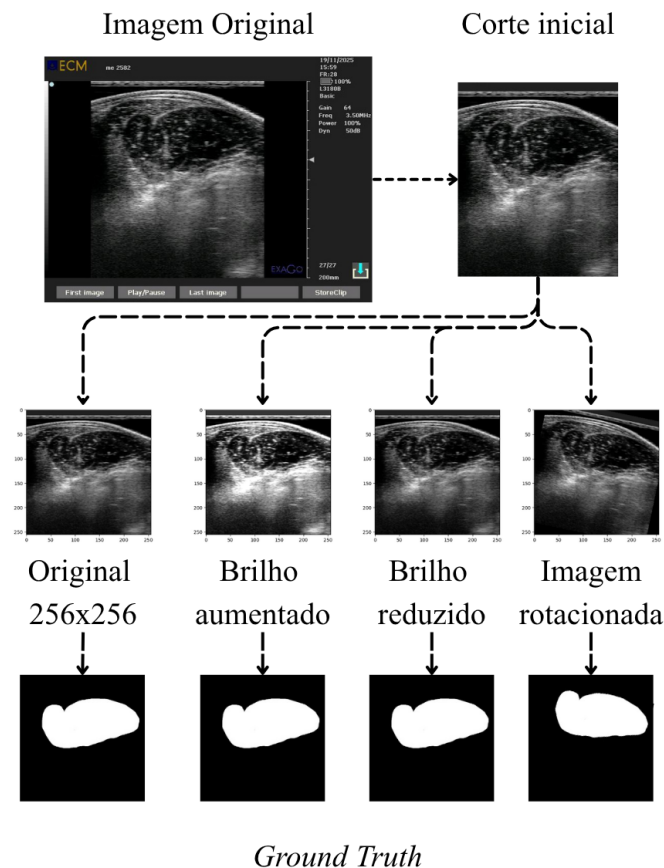
As imagens são divididas em conjuntos de treinamento (70%), validação (20%) e teste (10%). A biblioteca do Python, *label studio* (HUMANSIGNAL, 2020), é utilizada para criar o *ground truth* para as regiões correspondentes à AOL, EGL e EGG com base nas delimitações das regiões feitas pela especialista, sendo estas utilizadas como “rótulos”, ou seja, cada imagem do banco de dados tinha um *ground truth* correspondente. Devido à pequena quantidade de imagens disponíveis para o treinamento, é necessário aplicar técnicas de aumento de dados, do inglês *data augmentation*, para ampliar o conjunto de imagens utilizado no treinamento. Além disso, é realizado um recorte nas imagens para remover informações irrelevantes, que poderiam introduzir vieses no treinamento e comprometer o desempenho dos modelos. O processo de aumento de dados incluiu as seguintes transformações:

- Rotação: As imagens são rotacionadas em ângulos aleatórios dentro do intervalo de  $[-15^\circ, 15^\circ]$  para criar variações na orientação das imagens.
- Aumento e redução de brilho: Uma cópia de cada imagem é criada com o brilho aumentado em 50% e reduzido em 50% para simular diferentes condições de imagens.

Todas as modificações feitas no banco de dados são realizadas observando as características das imagens de ultrassom presentes, por exemplo, o aumento e a diminuição do brilho é baseado nas imagens mais claras e mais escuras do conjunto. Como cada imagem do conjunto de treinamento é modificada três vezes, o processo de aumento de dados resultou em um total de 372 imagens para o treinamento do modelo, 93 das imagens originais, com suas respectivas cópias modificadas, ou seja, 93 imagens com brilho aumentado, 93 imagens com brilho reduzido

e 93 imagens rotacionadas. Após o aumento de dados, as imagens são normalizadas, com os valores dos pixels ajustados para o intervalo  $[0, 1]$ , e redimensionadas para um tamanho fixo de  $256 \times 256$  pixels. As imagens, após o corte inicial, tinham dimensões de  $333 \times 403$ , a escolha de  $256 \times 256$  é feita para que cada imagem não fosse demasiadamente expandida e que não perdesse muita resolução. Outros testes são feitos para resoluções diferentes, como por exemplo,  $128 \times 128$  e  $512 \times 512$ , que resultaram em um desempenho inferior. Na Figura 4.3 é mostrado o processo de pré-processamento de uma imagem.

Figura 4.3: Exemplo do processo de pré-processamento das imagens de ultrassom.



Fonte: Elaborado pelo autor.

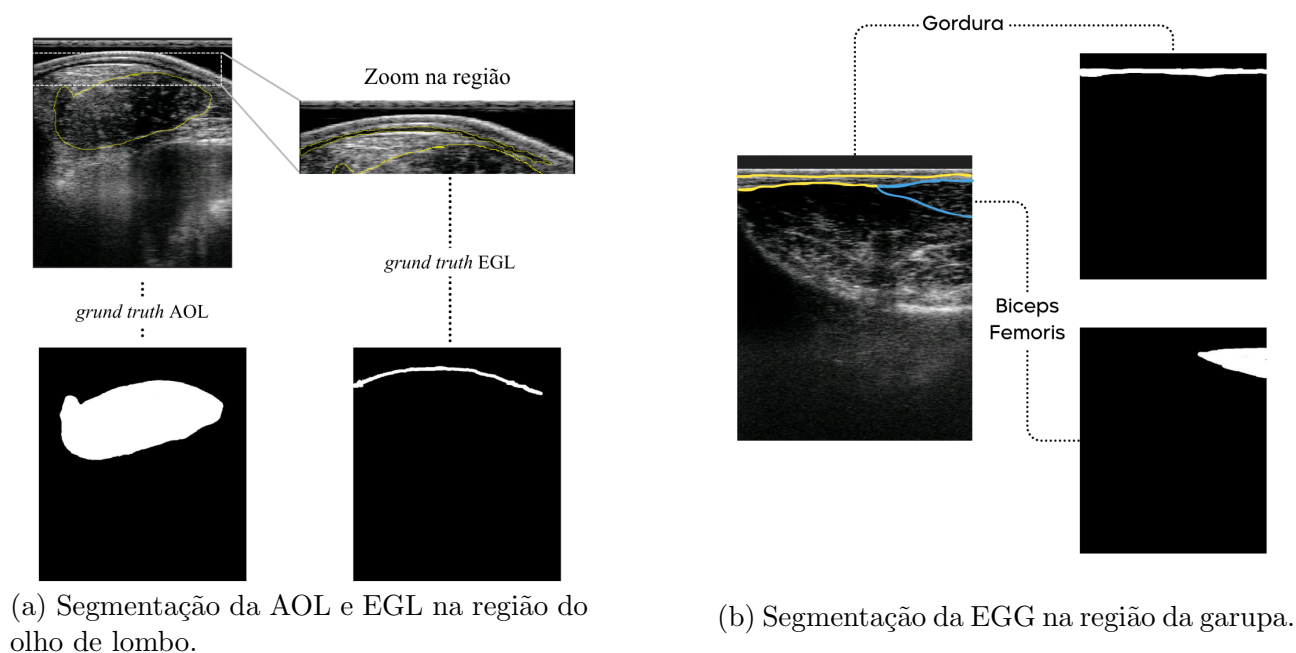
Na Figura 4.3, uma imagem é pré-processada e o *ground truth* da imagem original é adicionado a nova imagem criada. Observa-se que a imagem rotacionada mantém consistência com o *ground truth*, ou seja, o *ground truth* acompanha corretamente a rotação da imagem.

Cada medida extraída nas imagens de ultrassom (AOL, EGL e EGG) é representada por um valor real. Uma ideia inicial foi desenvolver uma CNN, descrita na Seção 1.3.2, que recebia como entrada as imagens de ultrassom e retornava como saída os valores numéricos correspondentes à AOL, EGL e EGG, ou seja, uma rede de regressão. No entanto, devido à quantidade baixa de imagens disponíveis para o treinamento do modelo, foi necessário realizar uma adaptação na abordagem inicial. Em vez de treinar uma CNN para prever os valores numéricos das medidas, optou-se, após análise literária, por criar *ground truth* para cada indicador (AOL, EGL e EGG)

a partir das imagens de ultrassom (MELO *et al.*, 2022). Com isso, os modelos recebem como entradas durante o treinamento, uma imagem de ultrassom junto com seu respectivo *ground truth* e retornam como saída máscaras de segmentação.

Para construir o *ground truth* da EGL, foi necessário demarcar toda a camada de gordura a partir da qual esse indicador é obtido. De modo análogo, para o EGG, foi preciso demarcar tanto a região de gordura quanto a região do músculo *Biceps Femoris*, gerando dois *ground truths* distintos: um correspondente à gordura e outro ao músculo, já que o indicador é obtido a partir da espessura da camada de gordura precisamente localizada no início do músculo. Na Figura 4.4 é mostrado um exemplo das máscaras de segmentação criadas para as imagens de ultrassom.

Figura 4.4: Exemplo das máscaras de segmentação criadas para as imagens de ultrassom.



Fonte: Elaborado pelo autor.

Na Figura 4.4(a) é mostrada a delimitação da região feita pela especialista e o respectivo *ground truth*. Na Figura 4.4(b) é mostrada a delimitação da região feita pela especialista, com cada uma tendo seu próprio *ground truth*.

Cada imagem de animal presente no banco de dados tinha um identificador numérico do animal. Animais com o primeiro dígito começando em 6, são animais da raça Nelore, enquanto animais com o primeiro dígito começando em 2, são animais da raça Caracu. Essa identificação serviu para concluir a organização do banco de dados e desenvolver os modelos como abordados na Seção 3.2.

## 4.2 Configuração dos Modelos

Esta seção descreve os procedimentos adotados para a comparação experimental entre os modelos propostos.

### 4.2.1 Modelos Comparados

Duas arquiteturas são comparadas para a segmentação das imagens de ultrassom: a U-Net Multitarefa e a U-Net Fuzzy. A U-Net Multitarefa incorpora a informação categórica da raça do animal, enquanto a U-Net Fuzzy mantém essa estrutura e adiciona o módulo de inferência fuzzy, conforme descrito no Capítulo 3. Em ambas as arquiteturas, a entrada do modelo durante o treinamento é composta pela imagem de ultrassom, o *ground truth* e pela codificação da raça do animal no formato *one-hot* (Nelore ou Caracu).

A comparação é realizada separadamente para cada um dos indicadores avaliados. Para a AOL e a EGL, é treinado um modelo de cada arquitetura. Para a EGG, devido à necessidade de identificar simultaneamente a camada de gordura e o músculo *Biceps Femoris*, são treinados dois modelos por arquitetura, um para cada estrutura anatômica. Assim, no total, são avaliados oito modelos.

Tabela 4.1: Quantidade de modelos desenvolvidos para cada medida.

| Tarefa de segmentação               | U-Net Multitarefa | U-Net Fuzzy |
|-------------------------------------|-------------------|-------------|
| AOL                                 | 1                 | 1           |
| EGL                                 | 1                 | 1           |
| EGG - camada de gordura             | 1                 | 1           |
| EGG - músculo <i>Biceps Femoris</i> | 1                 | 1           |
| Total                               | 4                 | 4           |

Fonte: Elaborado pelo autor.

Após o treinamento, a saída de cada modelo é uma máscara de segmentação e a partir desta máscara é possível extrair os indicadores correspondentes à AOL, EGL e EGG. Para a AOL, a medida é obtida a partir da contagem do número de pixels classificados como pertencentes à região de interesse (pixels brancos) e convertida para centímetros quadrados. Para EGL, os pixels pertencentes à região de interesse são contados e convertidos em milímetros, com isso é possível obter a largura da camada de gordura, a 3/4 desta largura é obtida a altura da camada de gordura, que é a medida da EGL. Para a EGG, como duas máscaras são criadas por dois modelos diferentes, quando uma imagem é enviada e processada pelos modelos, um retorna a máscara de segmentação da camada de gordura presente na Garupa e o outro retorna a máscara de segmentação do músculo *Biceps Femoris*. A partir dessas duas máscaras é possível obter a medida da EGG, primeiramente salvando a localização da coluna que se inicia o músculo *Biceps Femoris* (da esquerda para a direita) e calculando a espessura da gordura presente na mesma

coluna, ou seja, a espessura da camada de gordura na garupa que está no começo do músculo *Biceps Femoris*.

### 4.2.2 Configuração de Treinamento

Todos os modelos são treinados a partir das imagens previamente recortadas, normalizadas no intervalo  $[0, 1]$  e redimensionadas para  $256 \times 256$  pixels. As imagens destinadas ao treinamento são submetidas ao processo de aumento de dados descrito anteriormente, com o objetivo de ampliar a variabilidade do conjunto amostral e reduzir o risco de *overfitting*.

A configuração de treinamento é mantida, em sua maior parte, idêntica para as duas arquiteturas, a fim de tornar a comparação entre os modelos mais consistente. Em ambos os casos, é utilizado o otimizador Adam, (Seção 1.2.6), com taxa de aprendizado inicial de 0,001, tamanho de lote igual a 4, número máximo de 100 épocas e função de custo *Binary Cross-Entropy* (Seção 1.2.8). Além disso, são empregadas camadas de *Dropout* com taxa de 0,2, (Seção 1.9). O critério de *EarlyStopping*, (Seção 1.10), adota-se a convergência do treinamento baseado na métrica do coeficiente de Dice, equação (3.1), para o conjunto de validação, ou seja, caso o valor do coeficiente de Dice não melhore por 15 épocas consecutivas, o treinamento é interrompido e os melhores pesos do modelo são salvos.

Para a U-Net Fuzzy, além dessas configurações comuns, são definidos 4 antecedentes, dados pelo produto matricial entre o vetor de entrada do módulo fuzzy, multiplicado pela matriz de pesos e somados ao vetor de vieses (ver Seção 3.2.1). No domínio  $[-1, 1]$  dos antecedentes, são definidas 3 funções de pertinência do tipo Gaussiana. Essa configuração gera  $3^4 = 81$  regras. Para reduzir o efeito da aleatoriedade inerente ao processo de treinamento, cada modelo é treinado quatro vezes, portando, os resultados apresentados nas tabelas da próxima seção, são uma média dos valores obtidos nos 4 treinamentos. Na Tabela 4.2 é apresentado um resumo da configuração de treinamento adotada.

Tabela 4.2: Resumo da configuração de treinamento adotada nos experimentos.

| Parâmetro                      | Valor                       |
|--------------------------------|-----------------------------|
| Tamanho da imagem de entrada   | $256 \times 256$ pixels     |
| Normalização                   | intervalo $[0, 1]$          |
| Otimizador                     | Adam                        |
| Taxa de aprendizado inicial    | 0,001                       |
| Tamanho do lote                | 4                           |
| Número máximo de épocas        | 100                         |
| Função de custo                | <i>Binary Cross-Entropy</i> |
| <i>Dropout</i>                 | 0,2                         |
| Critério de parada             | <i>EarlyStopping</i>        |
| Número de execuções por modelo | 4                           |
| Antecedentes fuzzy             | 4                           |
| Funções de pertinência         | 3                           |
| Regras fuzzy                   | 81                          |

Fonte: Elaborado pelo autor.

### 4.2.3 Métricas de Avaliação

O desempenho dos modelos é analisado sob duas perspectivas complementares: a qualidade da máscara de segmentação gerada e a precisão dos indicadores extraídos a partir das máscaras preditas.

No contexto da segmentação, é adotado o coeficiente de Dice como principal métrica de avaliação, por quantificar diretamente a sobreposição entre a máscara predita pelo modelo e o *ground truth*. A acurácia é utilizada como métrica complementar, permitindo observar a proporção global de pixels classificados corretamente. Também são analisados o comportamento da função de custo ao longo das épocas e o número de épocas necessárias até a convergência, como indicadores da estabilidade do processo de treinamento.

No contexto da mensuração dos indicadores, é utilizado o MAE para quantificar, em unidade física, a magnitude média do erro entre os valores estimados e os valores de referência. Dessa forma, a combinação dessas métricas permite avaliar não apenas a capacidade dos modelos em segmentar corretamente as regiões de interesse, mas também sua utilidade prática na obtenção das medidas AOL, EGL e EGG.

### 4.2.4 Ambiente Computacional

Os experimentos são conduzidos em um notebook Asus TUF *Gaming* F15, equipado com processador Intel Core i7-12700H, 8 GB de memória RAM, unidade SSD de 512 GB e placa de vídeo NVIDIA GeForce RTX 3050 Mobile com 4 GB de memória VRAM. O sistema operacional utilizado foi o Linux Mint 22.3 “Zena” (LINUX MINT TEAM, 2026).

A implementação dos modelos é realizada na linguagem Python, versão 3.11.15 (PYTHON SOFTWARE FOUNDATION, 1990), com uso das bibliotecas TensorFlow 2.21.0, com suporte a GPU, e Keras 3.13.2 (GOOGLE LLC, 2015; KERAS TEAM, 2015). O desenvolvimento é realizado no ambiente Visual Studio Code (MICROSOFT, 2015). O treinamento das redes neurais é executado com aceleração por GPU, o que contribuiu para reduzir o tempo computacional necessário para a realização dos experimentos.

## 4.3 Resultados e Análise

Nesta seção são apresentados os resultados obtidos para cada um dos indicadores analisados, utilizando tanto a Unet Multitarefa, quanto a Unet Fuzzy. Embora haja indícios de *overfitting*, sendo a possível causa o tamanho pequeno do conjunto de dados de treinamento, os modelos mantiveram capacidade de generalização satisfatória, visto que, para dados não utilizados no treinamento, foram alcançados coeficiente de Dice de até 94,75% e acurácia de até 99%.

### 4.3.1 AOL

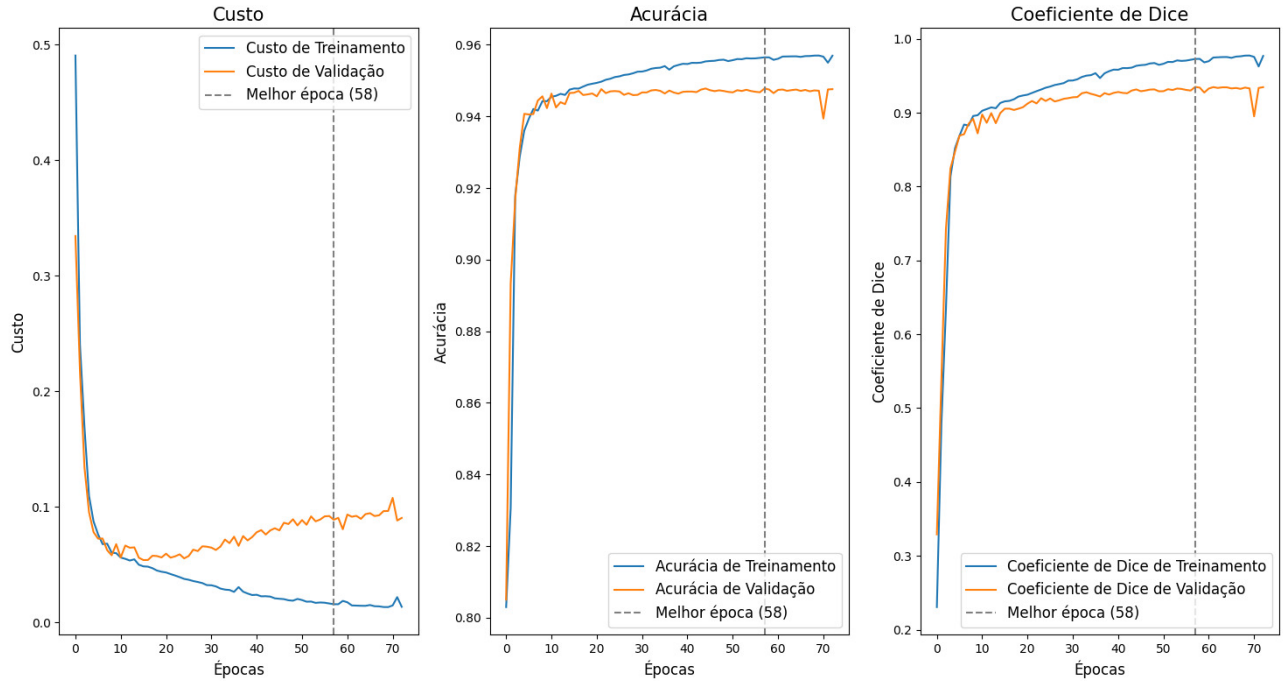
Nesta subseção são apresentados os resultados obtidos para a Área de Olho de Lombo (AOL), considerando tanto a qualidade da segmentação quanto a precisão das medidas numéricas extraídas a partir das máscaras preditas.

#### U-Net Multitarefa

Para a AOL, o modelo U-Net Multitarefa apresentou bom desempenho na tarefa de segmentação. No conjunto de validação, obteve coeficiente de Dice médio de 92,73% e acurácia média de 94,75%, indicando elevada sobreposição entre as máscaras preditas e as máscaras de referência. Em média, a convergência do treinamento ocorreu após 70 épocas. Em relação às medidas numéricas, o modelo alcançou MAE de 2,988 cm<sup>2</sup>. No conjunto de teste, o erro máximo observado é de 7,905 cm<sup>2</sup>, o erro mínimo é de 0,005 cm<sup>2</sup>, a média predita é de 39,880 cm<sup>2</sup> e a média real é de 40,559 cm<sup>2</sup>.

Na Figura 4.5, a função de custo apresenta redução até valores próximos de 0,09, enquanto as curvas de acurácia e Dice permanecem relativamente estáveis entre treinamento e validação. A linha tracejada indica a época em que são selecionados os melhores pesos por meio do critério de *Early Stopping*.

Figura 4.5: Gráficos de custo, Coeficiente de Dice e acurácia ao longo das épocas.



Fonte: Elaborado pelo autor.

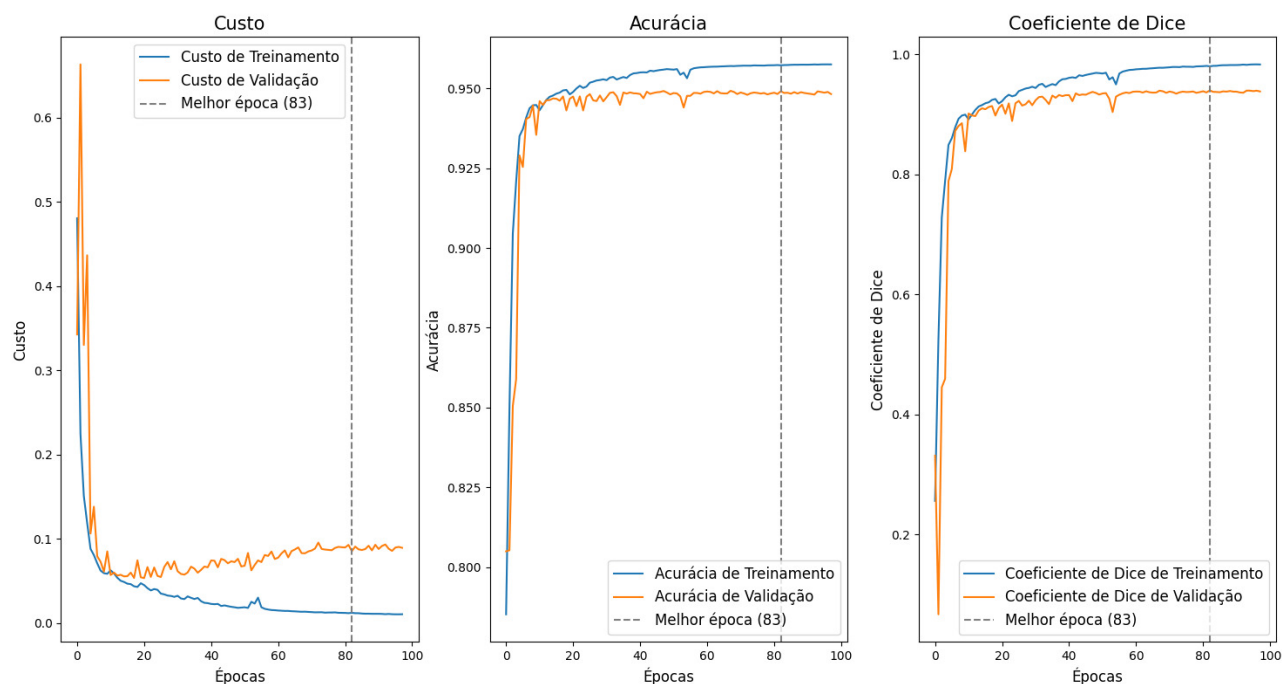
A linha tracejada na Figura 4.5 representa em qual época o modelo selecionou os melhores pesos por meio do critério de *EarlyStopping*, neste caso, o valor dos melhores pesos ocorreu na época de número 58, isso se deve ao fato de que os gráficos da Figura 4.5 são do primeiro dos quatro treinamentos realizados, por isso o valor de 58 difere do valor médio de 70. Assim também para os próximos gráficos apresentados nesta seção. Observa-se que o gráfico da função de custo apresenta valores decrescentes ao decorrer das épocas, isso é esperado tendo em vista que o modelo durante o treinamento está minimizando a função de custo. Os gráficos de acurácia e coeficiente de Dice apresentam valores crescentes ao decorrer das épocas, o que também é esperado, pois, como descrito na Subseção 3.3 o coeficiente de Dice próximo de 1 é o ideal, de forma semelhante para a acurácia.

## U-Net Fuzzy

Para a AOL, o modelo U-Net Fuzzy apresentou desempenho superior ao da U-Net Multitarefa na maior parte das métricas analisadas. No conjunto de validação, o modelo alcançou coeficiente de Dice médio de 93,29% e acurácia média de 95,15%. A convergência ocorreu, em média, após 89 épocas, valor superior ao observado para o modelo U-Net Multitarefa, isso se deve ao aumento de parâmetros introduzido pelo módulo fuzzy.

No que se refere à medida numérica, o modelo apresentou MAE de 2,312 cm<sup>2</sup>. No conjunto de teste, o erro máximo é de 6,644 cm<sup>2</sup>, o erro mínimo é de 0,225 cm<sup>2</sup>, a média predita é de 39,865 cm<sup>2</sup> com a média real sendo 40,559 cm<sup>2</sup>. Na Figura 4.6, observa-se que as curvas de custo, acurácia e Coeficiente de Dice apresentam maior oscilação ao longo do treinamento.

Figura 4.6: Gráficos de custo, Coeficiente de Dice e acurácia ao longo das épocas para o modelo de U-Net Fuzzy da AOL.



Fonte: Elaborado pelo autor.

A maior oscilação observada nas curvas de custo, acurácia e Coeficiente de Dice para o modelo U-Net Fuzzy pode ser atribuída também ao aumento da quantidade de parâmetros do modelo.

### Comparação entre os Modelos

Na Tabela 4.3 são resumidos os resultados obtidos para a AOL. De modo geral, o modelo U-Net Fuzzy superou o modelo U-Net Multitarefa nas principais métricas, apresentando maior coeficiente de Dice, maior acurácia e menor MAE. Esses resultados indicam que a integração do módulo fuzzy contribuiu positivamente tanto para a segmentação da região de interesse quanto para a extração dos indicadores numéricos da AOL. Por outro lado, o modelo fuzzy exigiu maior número de épocas até a convergência e apresentou maior instabilidade durante o treinamento.

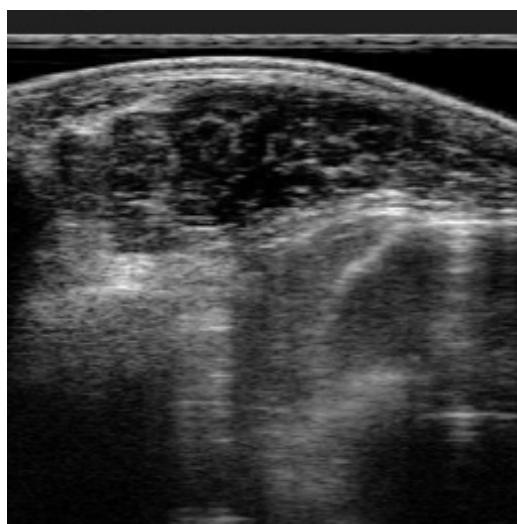
Na Figura 4.7 é apresentada uma comparação visual entre o *ground truth* e as máscaras de segmentação geradas pelos modelos para a mesma imagem de ultrassom. Observa-se que ambas as máscaras ficam sobrepostas à referência, com diferenças sutis, o que é coerente com os valores elevados de coeficiente de Dice obtidos por ambos os modelos. As máscaras de segmentação são modificadas para serem sobrepostas à imagem original, a fim de facilitar a visualização e comparação entre os modelos.

Tabela 4.3: Resumo dos resultados obtidos para a AOL, comparando o modelo de U-Net Multitarefa com o modelo de U-Net Fuzzy.

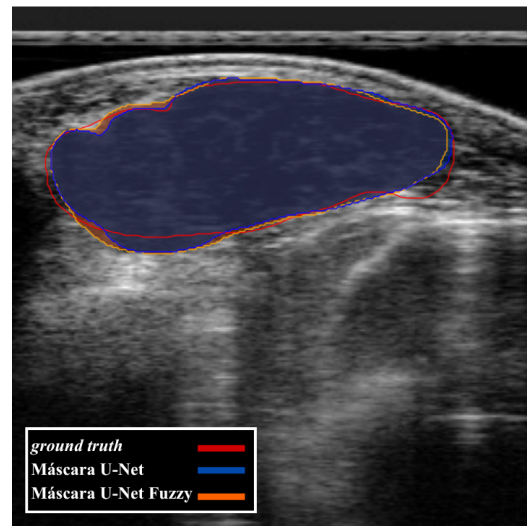
| Métrica              | U-Net Multitarefa      | U-Net Fuzzy            |
|----------------------|------------------------|------------------------|
| Coefficiente de Dice | 92,73%                 | 93,29%                 |
| Acurácia             | 94,75%                 | 95,15%                 |
| Número de Épocas     | 70                     | 89                     |
| MAE                  | 2,988 cm <sup>2</sup>  | 2,312 cm <sup>2</sup>  |
| Erro Máximo          | 7,905 cm <sup>2</sup>  | 6,644 cm <sup>2</sup>  |
| Erro Mínimo          | 0,005 cm <sup>2</sup>  | 0,225 cm <sup>2</sup>  |
| Área Média Predita   | 39,880 cm <sup>2</sup> | 39,865 cm <sup>2</sup> |
| Área Média Real      | 40,559 cm <sup>2</sup> | 40,559 cm <sup>2</sup> |

Fonte: Elaborado pelo autor.

Figura 4.7: Comparação entre as máscaras de segmentação da AOL.



(a) Imagem de ultrassom original.



(b) *ground truth* e máscaras de segmentação geradas pelos modelos.

Fonte: Elaborado pelo autor.

Na Figura 4.7(a) é apresentada a imagem de ultrassom original. Na Figura 4.7(b) é mostrado a o *ground truth* em vermelho, a máscara de segmentação gerada pelo modelo de U-Net Multitarefa em azul e a máscara de segmentação gerada pelo modelo de U-Net Fuzzy em laranja, todas sobrepostas sobre a imagem original.

### 4.3.2 EGL

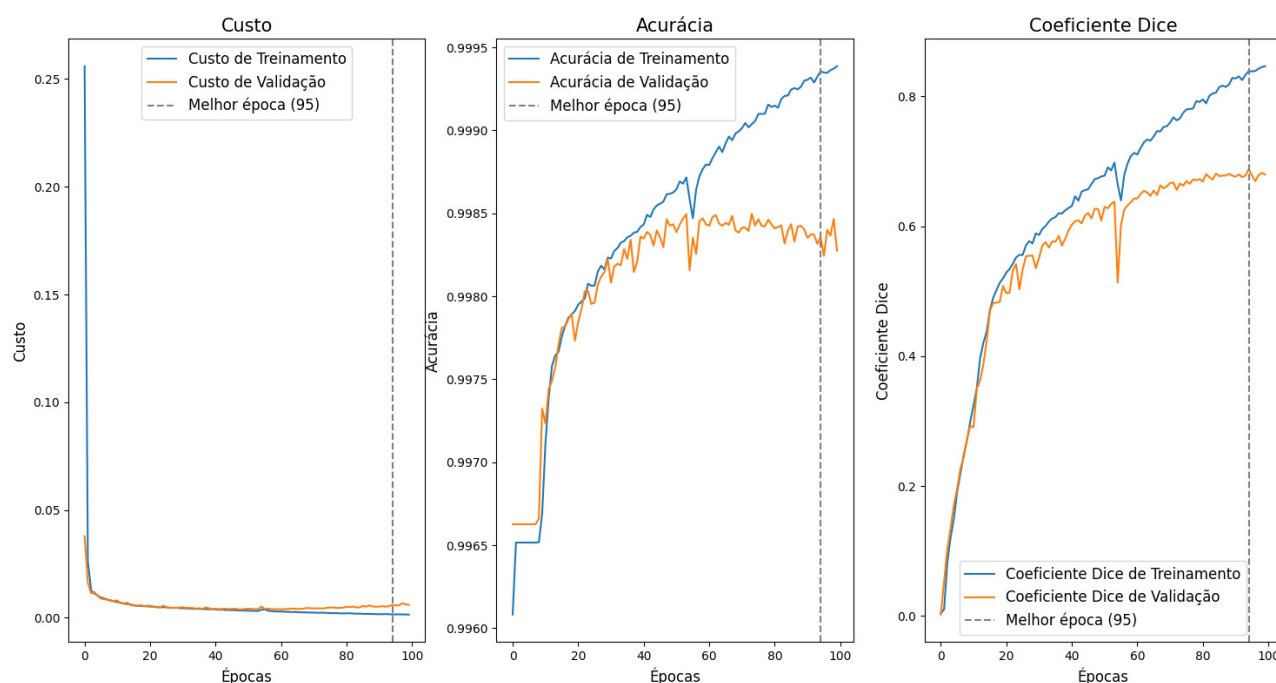
Nesta subseção são apresentados os resultados obtidos para a EGL. Em comparação com a AOL, trata-se de uma tarefa mais desafiadora, pois a camada de gordura apresenta menor espessura e menor contraste em relação ao fundo da imagem.

## U-Net Multitarefa

Para a EGL, o modelo U-Net Multitarefa, no conjunto de validação, obteve coeficiente de Dice médio de 68,10% e acurácia média de 99,80%. Apesar da acurácia elevada, o coeficiente de Dice é consideravelmente inferior ao observado para a AOL, o que indica maior dificuldade na delimitação da camada de gordura. Esse comportamento pode ser explicado pelo fato da região segmentada ser estreita e ocupar uma pequena fração da imagem, tornando a tarefa mais sensível a pequenas variações de contorno. Em média, a convergência ocorreu após 88 épocas.

No que se refere à medida numérica, o modelo apresentou MAE de 0,018 mm. No conjunto de teste, o erro máximo observado é de 0,48 mm, o erro mínimo é de 0,01 mm, e as médias predita e real são ambas de 0,134 mm. Na Figura 4.8 é mostrado o comportamento das curvas de custo, coeficiente de Dice e acurácia ao longo do treinamento.

Figura 4.8: Gráficos de custo, Coeficiente de Dice e acurácia ao longo das épocas para o modelo U-Net Multitarefa da EGL.



Fonte: Elaborado pelo autor.

Observa-se que a função de custo converge rapidamente para valores muito baixos, o que não implica necessariamente boa segmentação da camada de gordura. Este efeito ocorre porque a maior parte dos pixels pertencem ao fundo da imagem, o que favorece soluções triviais e explica, em parte, a discrepância entre a acurácia elevada e o coeficiente de Dice mais baixo.

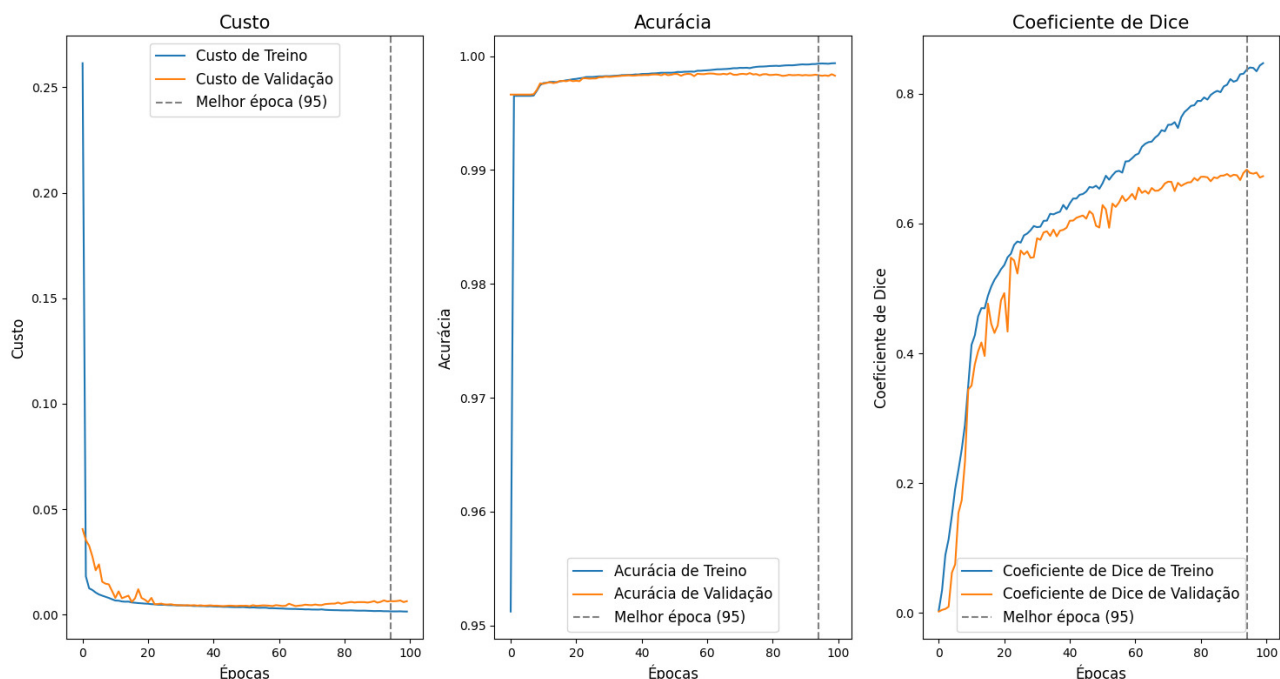
## U-Net Fuzzy

Para a EGL, o modelo U-Net Fuzzy apresentou desempenho semelhante ao da U-Net Multitarefa na tarefa de segmentação, com pequenas diferenças entre as métricas analisadas. O modelo alcançou coeficiente de Dice médio de 68,28% e acurácia média de 99,95% para o conjunto de

validação. A convergência ocorreu, em média, após 94 épocas.

Em relação à medida numérica, o modelo apresentou MAE de 0,015 mm. No conjunto de teste, o erro máximo é de 0,54 mm, o erro mínimo é de 0,01 mm, e as médias predita e real são de 0,134 mm. Na Figura 4.9, observa-se que as curvas de custo, coeficiente de Dice e acurácia apresentam maior oscilação ao longo do treinamento, padrão semelhante ao observado para a AOL. Esse comportamento pode estar associado à maior complexidade introduzida pelo módulo fuzzy.

Figura 4.9: Gráficos de custo, Coeficiente de Dice e acurácia ao longo das épocas para o modelo U-Net Fuzzy da EGL.



Fonte: Elaborado pelo autor.

É possível observar em ambos os gráficos (Figura 4.8 e Figura 4.9) uma tendência de overfitting a partir de aproximadamente 60 épocas, sendo as possíveis causas o tamanho do conjunto de treinamento e a complexidade da região. Apesar disso, ambos os modelos mantiveram capacidade de generalização satisfatória.

### Comparação entre os Modelos

Na Tabela 4.4 são resumidos os resultados obtidos para a EGL. De forma geral, as diferenças entre os dois modelos são pequenas. O modelo U-Net Fuzzy apresentou ligeira vantagem nas métricas de segmentação e no MAE.

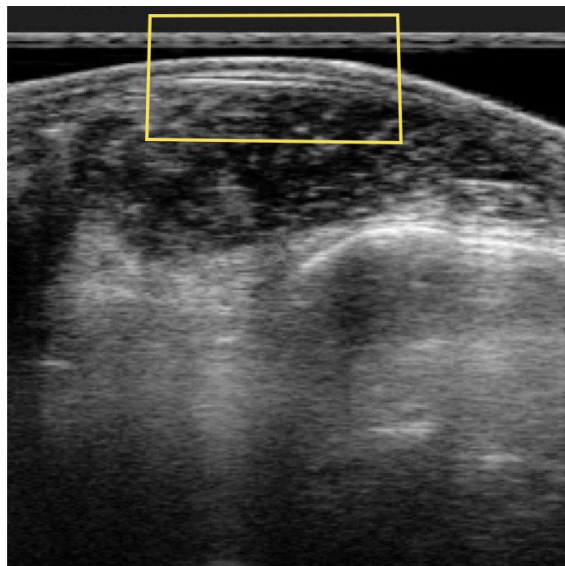
Tabela 4.4: Resumo dos resultados obtidos para a EGL, comparando o modelo U-Net Multitarefa com o modelo U-Net Fuzzy.

| Métrica              | U-Net Multitarefa | U-Net Fuzzy |
|----------------------|-------------------|-------------|
| Coefficiente de Dice | 68,10%            | 68,28%      |
| Acurácia             | 99,85%            | 99,95%      |
| Número de Épocas     | 88                | 94          |
| MAE                  | 0,018 mm          | 0,015 mm    |
| Erro Máximo          | 0,48 mm           | 0,54 mm     |
| Erro Mínimo          | 0,01 mm           | 0,01 mm     |
| Média Predita        | 0,134 mm          | 0,134 mm    |
| Média Real           | 0,134 mm          | 0,134 mm    |

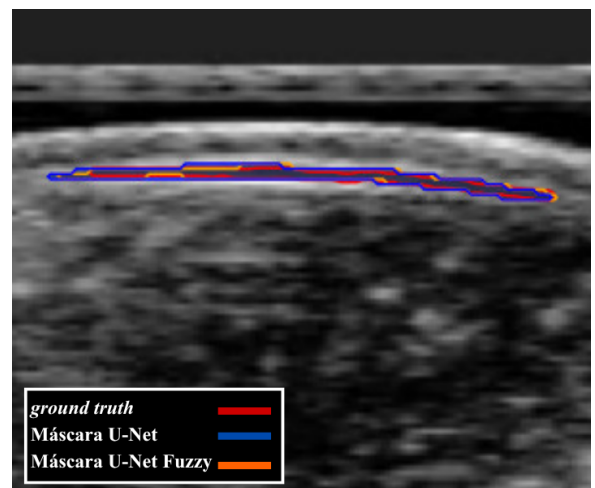
Fonte: Elaborado pelo autor.

Na Figura 4.10 é apresentada uma comparação visual entre o *ground truth* e as máscaras de segmentação geradas pelos modelos. Observa-se que ambas estão sobrepostas ao *ground truth*, com diferenças discretas. O coeficiente de Dice relativamente menor para os dois modelos decorre do fato da região segmentada ser pequena, de modo que pequenas diferenças no contorno resultam em variações relativamente grandes nessa métrica.

Figura 4.10: Comparação entre as máscaras de segmentação.



(a) Imagem de ultrassom original.



(b) *ground truth* e máscaras de segmentação geradas pelos modelos.

Fonte: Elaborado pelo autor.

Na Figura 4.10(a) é apresentada a imagem de ultrassom original com um retângulo amarelo. Na Figura 4.10(b) é mostrado o *ground truth* em vermelho, a máscara de segmentação gerada pelo modelo U-Net Multitarefa em azul e a máscara de segmentação gerada pelo modelo U-Net Fuzzy em laranja, todas sobrepostas sobre a imagem original com um zoom na região do

retângulo amarelo da Figura 4.10(a).

### 4.3.3 EGG

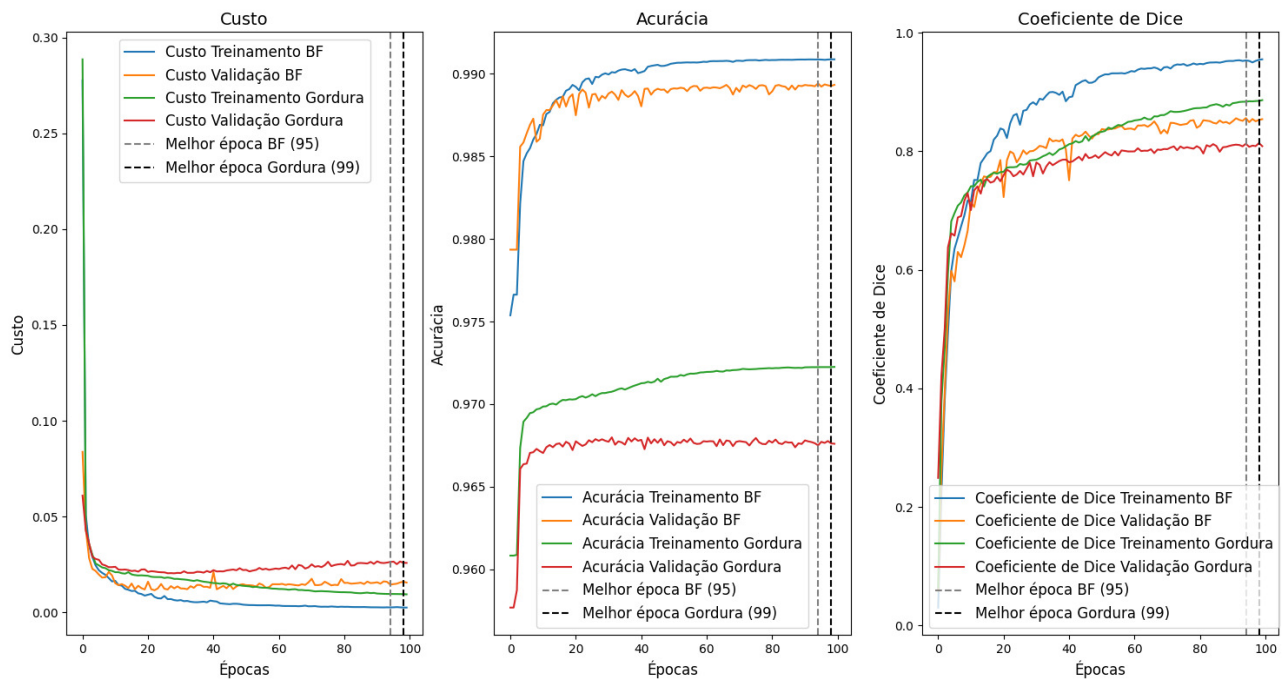
Nesta subseção são apresentados os resultados obtidos para a EGG. Diferentemente dos indicadores anteriores, a obtenção da EGG depende de duas segmentações distintas: uma para a camada de gordura e outra para o músculo *Biceps Femoris*. Assim, eventuais erros em qualquer uma dessas etapas podem se propagar para a medida final.

#### U-Net Multitarefa

Para a segmentação do músculo *Biceps Femoris*, o modelo U-Net Multitarefa apresentou coeficiente de Dice médio de 82,74% e 79,84% para a segmentação da região de gordura. As acurácias médias correspondentes são de 98,75% e 96,87%, respectivamente. A convergência ocorreu, em média, após 31 épocas para o músculo *Biceps Femoris* e 61 épocas para a região de gordura.

Em relação à medida numérica, o modelo apresentou MAE de 0,7 mm. No conjunto de teste, o erro máximo observado é de 2,5 mm, o erro mínimo é de 0,001 mm, a média predita é de 4,915 mm e a média real é de 4,781 mm.

Figura 4.11: Gráficos de custo, Coeficiente de Dice e acurácia ao longo das épocas para os modelos U-Net Multitarefa da EGG.



Fonte: Elaborado pelo autor.

A abreviação “BF” presente na legenda da Figura 4.11 refere-se ao músculo *Biceps Femoris*. Observa-se que os gráficos para a segmentação do músculo *Biceps Femoris* apresentam maior oscilação, o que pode ser atribuído ao tamanho relativamente menor da região segmentada, o

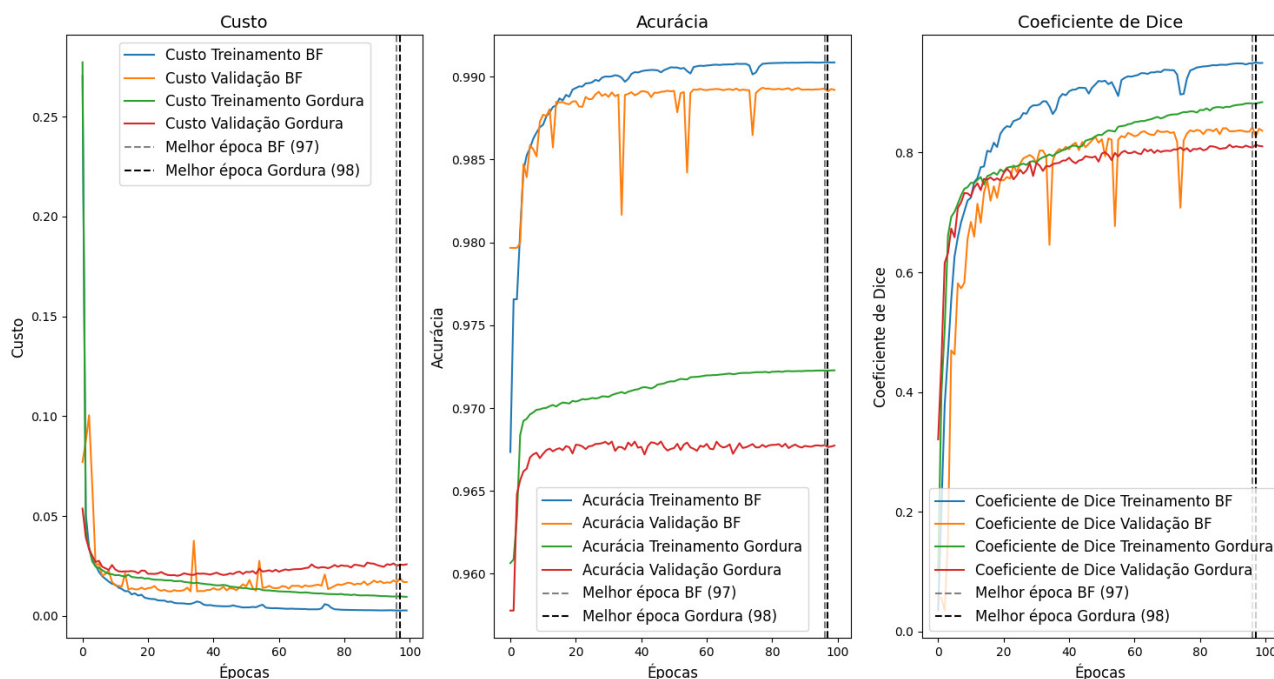
que torna a tarefa mais sensível a pequenas variações. Os gráficos da Figura 4.11 apresentam o compartimento de ambas as medidas, entretanto, o treinamento é realizado separadamente, com os valores de cada época sendo salvos e apresentados em conjunto posteriormente.

## U-Net Fuzzy

O modelo U-Net Fuzzy apresentou coeficiente de Dice médio de 84,60% para a segmentação do músculo *Biceps Femoris* e 79,99% para a segmentação da região de gordura. As acurácias médias correspondentes são de 98,60% e 96,88%, respectivamente. A convergência ocorreu, em média, após 90 épocas para o músculo e 92 épocas para a região de gordura.

Para a medida numérica da EGG, o modelo apresentou MAE de 0,8 mm. No conjunto de teste, o erro máximo observado é de 2,1 mm, o erro mínimo é de 0,001 mm, a média predita é de 4,974 mm e a média real é de 4,781 mm. Na Figura 4.12 é mostrado o comportamento dos gráficos de custo, coeficiente de Dice e acurácia ao longo do treinamento para o modelo U-Net Fuzzy que calcula a segmentação do músculo *Biceps Femoris* junto com os gráficos para o modelo U-Net Fuzzy que calcula a segmentação da região de gordura, ambos utilizados para a obtenção da EGG.

Figura 4.12: Gráficos de custo, Coeficiente de Dice e acurácia ao longo das épocas para os modelos U-Net Fuzzy da EGG.



Fonte: Elaborado pelo autor.

Observa-se que, assim como nos resultados anteriores, os gráficos da Figura 4.12 apresentam maior oscilação.

## Comparação entre os Modelos

Na Tabela 4.5 são resumidos os resultados obtidos para a EGG. Em termos de segmentação, os dois modelos apresentaram desempenho semelhante, com leve vantagem da U-Net Fuzzy nos valores de coeficiente de Dice. Entretanto, ao analisar a medida final da EGG, o modelo U-Net Multitarefa apresentou melhor desempenho, com menor MAE.

Tabela 4.5: Resultados das métricas para a EGG.

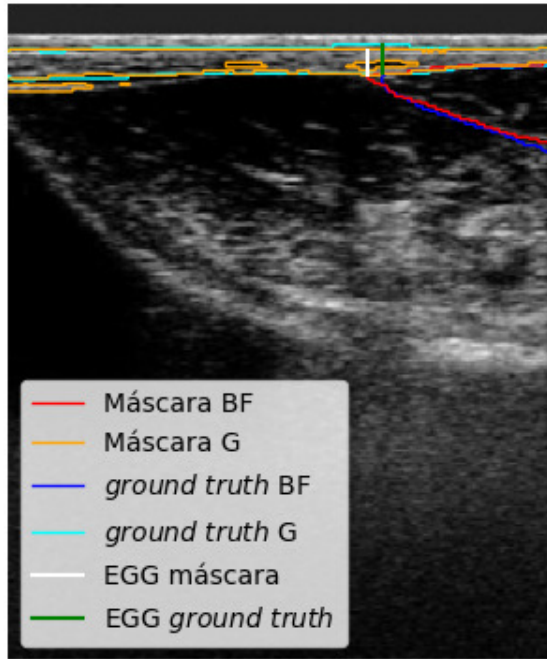
| Métrica             | U-Net Multitarefa              | U-Net Fuzzy                    |
|---------------------|--------------------------------|--------------------------------|
| Coeficiente de Dice | <i>Biceps Femoris</i> : 82,74% | <i>Biceps Femoris</i> : 84,60% |
|                     | Gordura: 79,84%                | Gordura: 79,99%                |
| Acurácia            | <i>Biceps Femoris</i> : 98,75% | <i>Biceps Femoris</i> : 98,60% |
|                     | Gordura: 96,87%                | Gordura: 96,88%                |
| Número de Épocas    | <i>Biceps Femoris</i> : 31     | <i>Biceps Femoris</i> : 90     |
|                     | Gordura: 61                    | Gordura: 92                    |
| MAE                 | 0,7 mm                         | 0,8 mm                         |
| Erro Máximo         | 2,5 mm                         | 2,1 mm                         |
| Erro Mínimo         | 0,001 mm                       | 0,001 mm                       |
| Média Preditada     | 4,915 mm                       | 4,974 mm                       |
| Média Real          | 4,781 mm                       | 4,781 mm                       |

Fonte: Elaborado pelo autor.

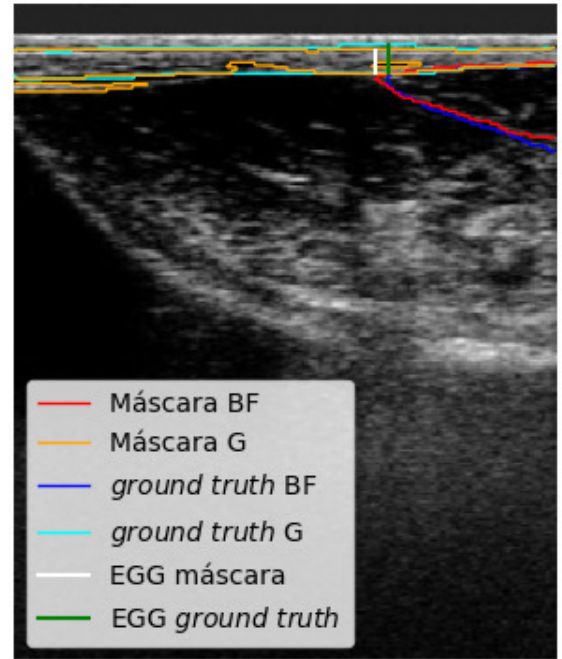
Na Figura 4.13 são apresentadas uma comparação visual entre o modelo U-Net Multitarefa, Figura 4.13(a), e o modelo U-Net Fuzzy, Figura 4.13(b), para a mesma imagem de ultrassom, em que a legenda “Máscara BF” refere-se à máscara de segmentação do músculo *Biceps Femoris* e a legenda “Máscara G” refere-se à máscara de segmentação da região de gordura, de forma similar para as legendas que começam com “*ground truth*”.

As máscaras de segmentação e o *ground truth* estão em formato de delimitação, ou seja, o interior da máscara não está sendo mostrado, isso é escolhido para facilitar a visualização da região segmentada. A cor vermelha representa a máscara de segmentação do músculo *Biceps Femoris*, a cor laranja representa a máscara de segmentação da região de gordura, a cor azul escuro representa o *ground truth* do músculo *Biceps Femoris*, a cor azul claro representa o *ground truth* da região de gordura, cor branca representa o local em que é obtido o valor do indicador da EGG para as máscaras de segmentação e a cor verde representa o local em que é obtido o valor do indicador da EGG para o *ground truth*. Esse resultado sugere que, para a EGG, pequenas melhorias na segmentação isolada das estruturas anatômicas não se traduziram necessariamente em maior precisão da medida numérica final. Como a EGG depende da combinação de duas segmentações e da localização exata do início do músculo *Biceps Femoris*, a propagação de pequenos erros compromete de forma mais acentuada a medida final.

Figura 4.13: Comparação entre as máscaras de segmentação para a EGG.



(a) Máscaras de segmentação geradas pelo modelo U-Net Multitarefa.



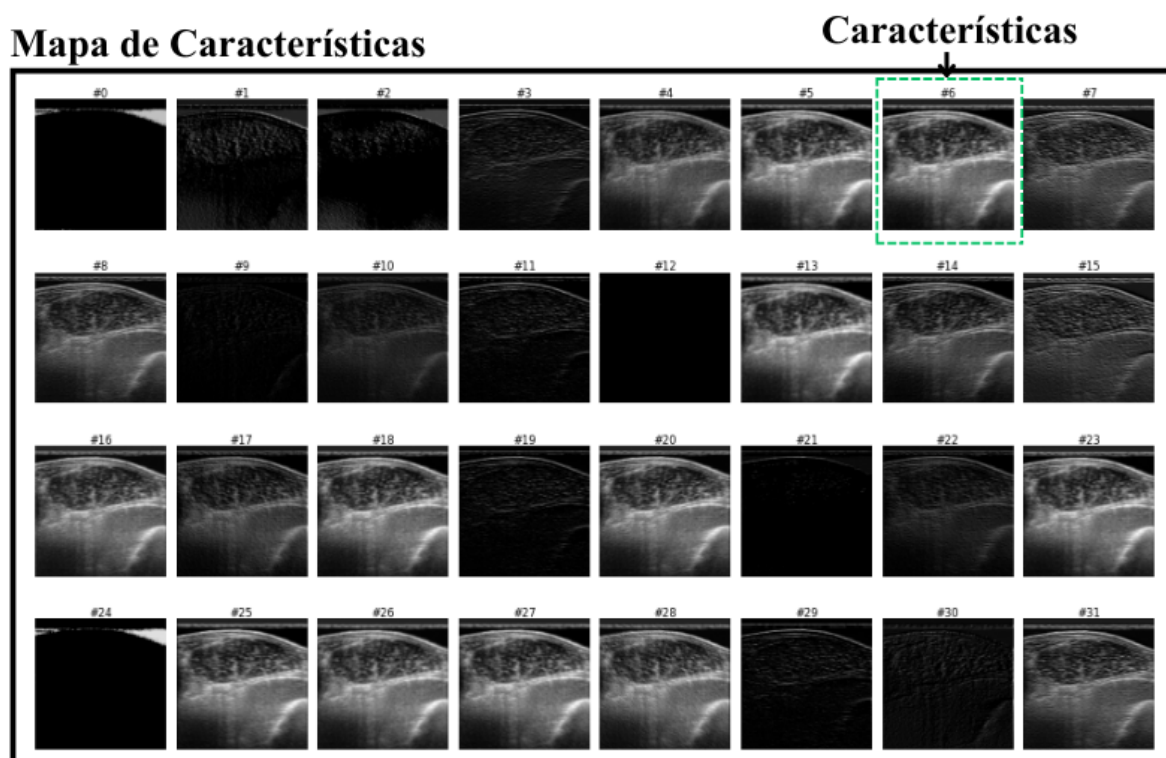
(b) Máscaras de segmentação geradas pelo modelo U-Net Fuzzy.

Fonte: Elaborado pelo autor.

## 4.4 Análise Geral dos Resultados

De forma geral, os resultados obtidos para os três indicadores analisados (AOL, EGL e EGG) indicam que a integração do módulo de inferência fuzzy na arquitetura de U-Net proporcionou melhorias na qualidade da segmentação, especialmente para a AOL. Como desenvolvido anteriormente, a arquitetura U-Net Multitarefa/U-Net Fuzzy é formada por camadas de convolução, camadas de pooling e camadas de convolução transposta, além de conexões de salto entre as camadas correspondentes. Quando a imagem é processada pela primeira camada de convolução, a imagem resultante passa a ser conhecida como mapa de características, que geralmente tem uma quantidade de canais maior que a imagem de entrada, estes canais podem ser chamados de características. Para entender o motivo do nome, é necessário visualizar o que acontece em cada etapa de processamento das arquiteturas. Na Figura 4.14 é mostrado o processo de extração de características e geração do mapa de características na primeira etapa de convolução dos modelos, para o indicador de AOL. Esta etapa recebe uma imagem ( $256 \times 256 \times 1$ ) e transforma em um mapa de características ( $256 \times 256 \times 32$ ), ou seja, a imagem é processada por 32 *kernels* com 1 canal, resultando no mapa de características com 32 canais. Para esta análise, os canais nos mapas de características são chamados de características. Alguns mapas de características presentes nesta seção não apresentam todas as características extraídas, isso é feito para facilitar a visualização, pois o mapa de características pode conter até 256 características, o que dificultaria a visualização.

Figura 4.14: Processo de extração de características e geração do mapa de características na primeira etapa de convolução da U-Net.

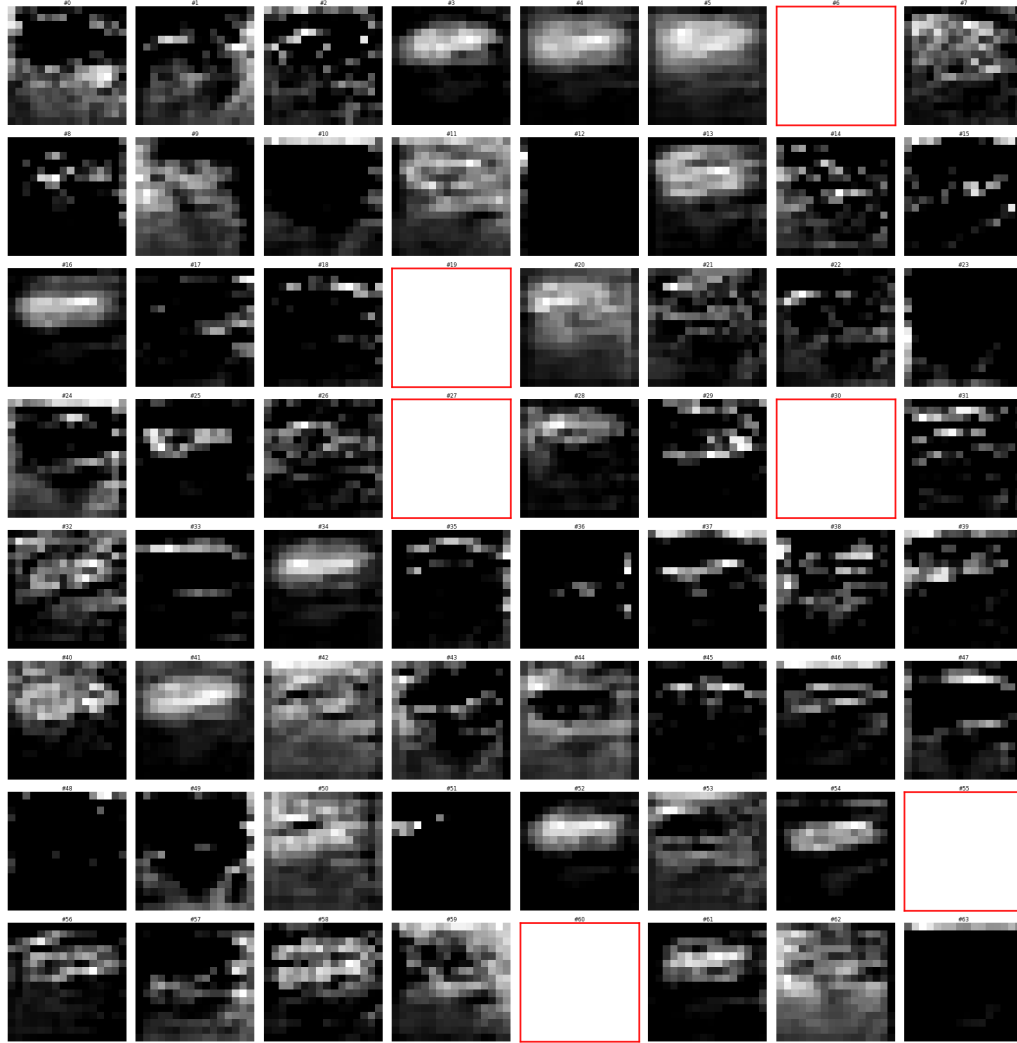


Fonte: Elaborado pelo autor.

Na Figura 4.14 é mostrada como é gerado o mapa de características na primeira etapa de convolução, após a época 60, isso implica que a rede já passou por um processo de otimização dos pesos e as características extraídas não são aleatórias, o que acontece em épocas iniciais, mas sim características relevantes para a rede produzir a máscara de segmentação.

Em cada etapa de convolução nem todas as características são relevantes para a rede criar a máscara de segmentação da imagem, observa-se por exemplo, que na última etapa de convolução do *bottleneck* (antes de ser enviado para a camada multitarefa ou para o módulo fuzzy), presente na Figura 4.15, algumas características não são ativadas pela rede, ou seja, apresentam valores próximos de zero, o que não significa que esta é irrelevante, mas sim que, para a imagem de entrada, a rede não encontrou informações relevantes para ativá-la. As características que não são ativadas estão brancas e com a borda vermelha para facilitar a visualização.

Figura 4.15: Mapa de características na última etapa de convolução do *bottleneck* da U-Net.

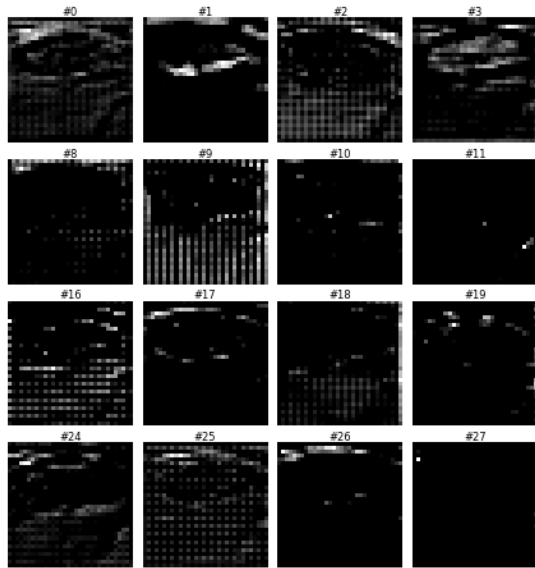


Fonte: Elaborado pelo autor.

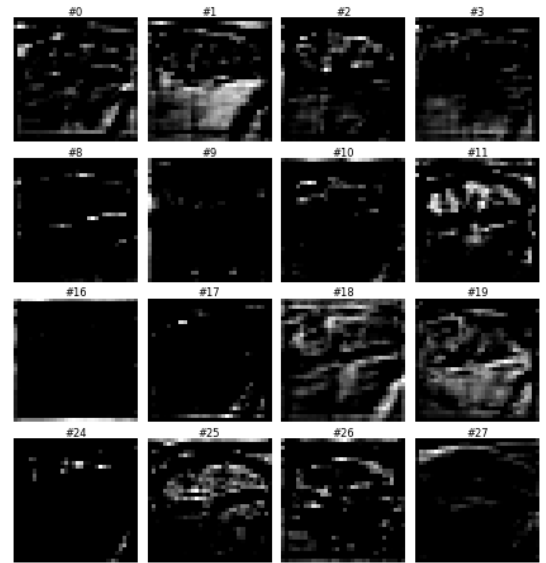
Observa-se que em algumas subfiguras da Figura 4.15 a rede extrai características semelhantes ao *ground truth* da AOL. Nesta etapa de convolução o mapa de características tem dimensões de  $16 \times 16 \times 256$ , ou seja, o mapa de características anterior é processado por 256 *kernels* de dimensões  $3 \times 3 \times 256$ , resultando em 256 características, cada uma com dimensões de  $16 \times 16$ . Para ser possível visualizar essas características na Figura 4.15 são selecionadas apenas 64 características.

É no início do *decoder* que acontece a principal diferença entre os modelos. A U-Net Fuzzy apresenta o módulo de inferência fuzzy que recebe o mapa de características da última etapa de convolução do *bottleneck*, após a camada de multitarefa, e processa este mapa para gerar um novo mapa de características, que é enviado para o *decoder*. Na Figura 4.16 é possível visualizar as diferenças entre os mapas de características gerados, com o mapa na Figura 4.16(a) passando pelo módulo de inferência fuzzy e o mapa na Figura 4.16(b) não. Estes mapas são gerados no primeiro bloco do *decoder*.

Figura 4.16: Mapa de características gerado com o módulo de inferência fuzzy e mapa de características gerado sem o módulo de inferência fuzzy.



(a) Mapa de características gerado pelo módulo de inferência fuzzy.



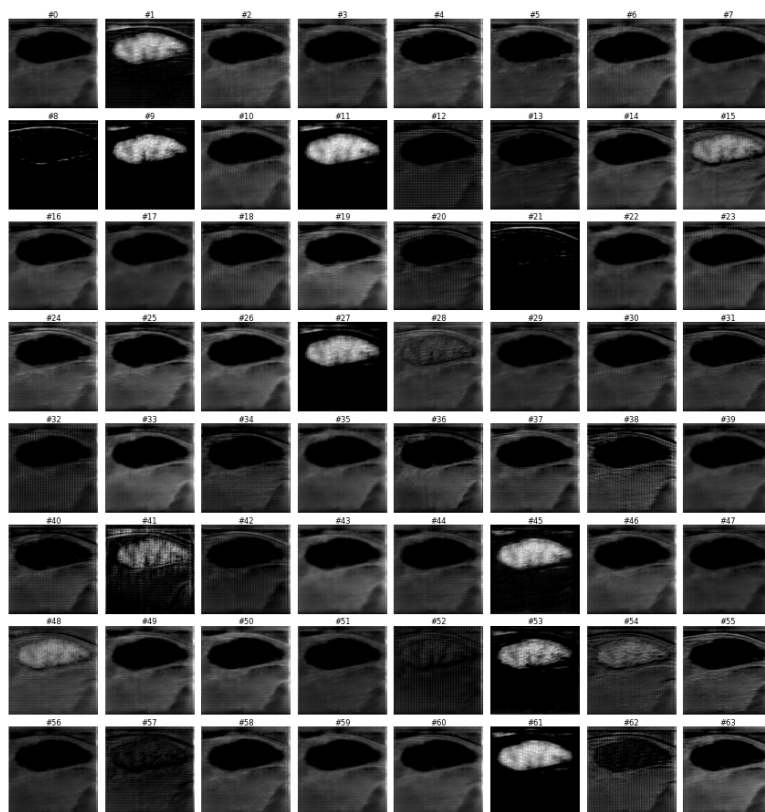
(b) Mapa de características sem o módulo de inferência fuzzy.

Fonte: Elaborado pelo autor.

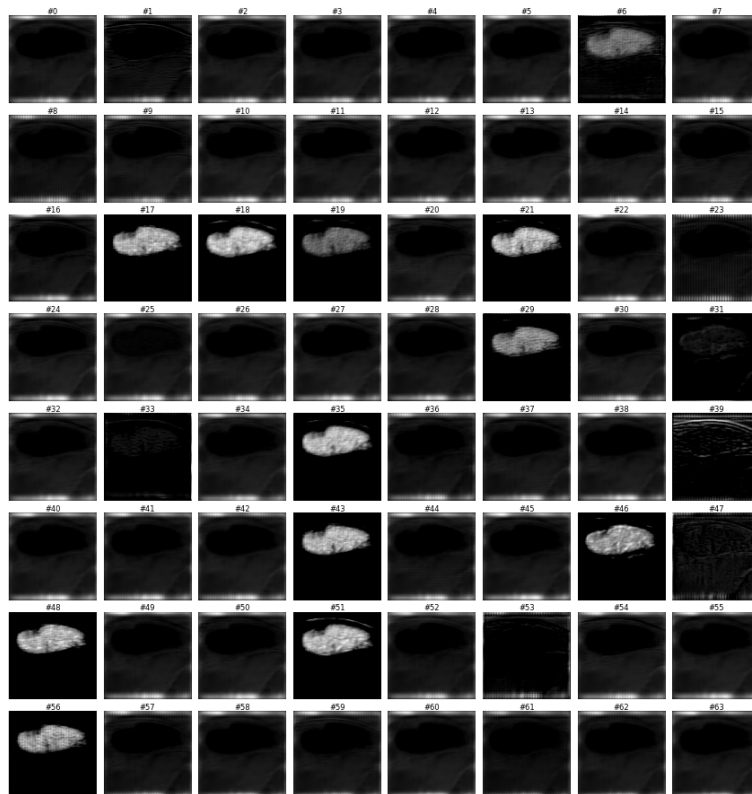
Na Figura 4.16(a) é possível notar um padrão quadriculado em algumas características, isso acontece devido a adição do módulo fuzzy antes da convolução transposta, Seção 1.3.9, e como foi escolhido plotar os mapas de características, com cada pixel passando por uma normalização para o intervalo  $[0, 1]$  ressaltando ainda mais o padrão quadriculado.

Como os modelos diferem na saída do *bottleneck*, o *decoder* de cada arquitetura passa a reconstruir a imagem de maneira distinta. Este fato pode ser observado na Figura 4.17(a), em que são apresentados os mapas de características gerados no terceiro bloco do *decoder* da U-Net Fuzzy e da U-Net Multitarefa. No caso da U-Net Fuzzy, o mapa de características evidencia com maior nitidez a separação entre a região correspondente à AOL e as demais regiões da imagem de ultrassom. No modelo U-Net Multitarefa, o mapa de características apresenta um padrão mais homogêneo com menor distinção entre a região de AOL e o restante da imagem, como é possível ser observado na Figura 4.17(b).

Figura 4.17: Mapa de características gerado no terceiro bloco do *decoder* da U-Net Fuzzy e da U-Net Multitarefa.



(a) Mapa de características do *decoder* da U-Net Fuzzy.



(b) Mapa de características do *decoder* da U-Net Multitarefa.

Fonte: Elaborado pelo autor.

Como é possível perceber nos resultados apresentados para a AOL, isso se traduz em uma máscara de segmentação mais precisa da região de interesse, o que impacta positivamente na extração da medida numérica.

## 4.5 Aplicativo Computacional

Com o objetivo de facilitar a utilização dos modelos desenvolvidos, foi criado um aplicativo computacional com interface gráfica, utilizando a biblioteca *Tkinter* do Python. O aplicativo permite que o usuário selecione uma imagem de ultrassom, escolha o modelo desejado (U-Net Multitarefa ou U-Net Fuzzy) e visualize a máscara de segmentação gerada, bem como os valores numéricos dos indicadores (AOL, EGL e EGG). O aplicativo é dividido em três etapas principais.

- A primeira etapa é destinada à seleção do indicador que é medido (AOL, EGL ou EGG) e o modelo que é utilizado para a segmentação (U-Net Multitarefa ou U-Net Fuzzy). Na Figura 4.18 é mostrado a tela inicial do aplicativo, em que é selecionado o indicador AOL e o modelo U-Net Fuzzy.

Figura 4.18: Tela inicial do aplicativo para seleção do indicador e modelo.

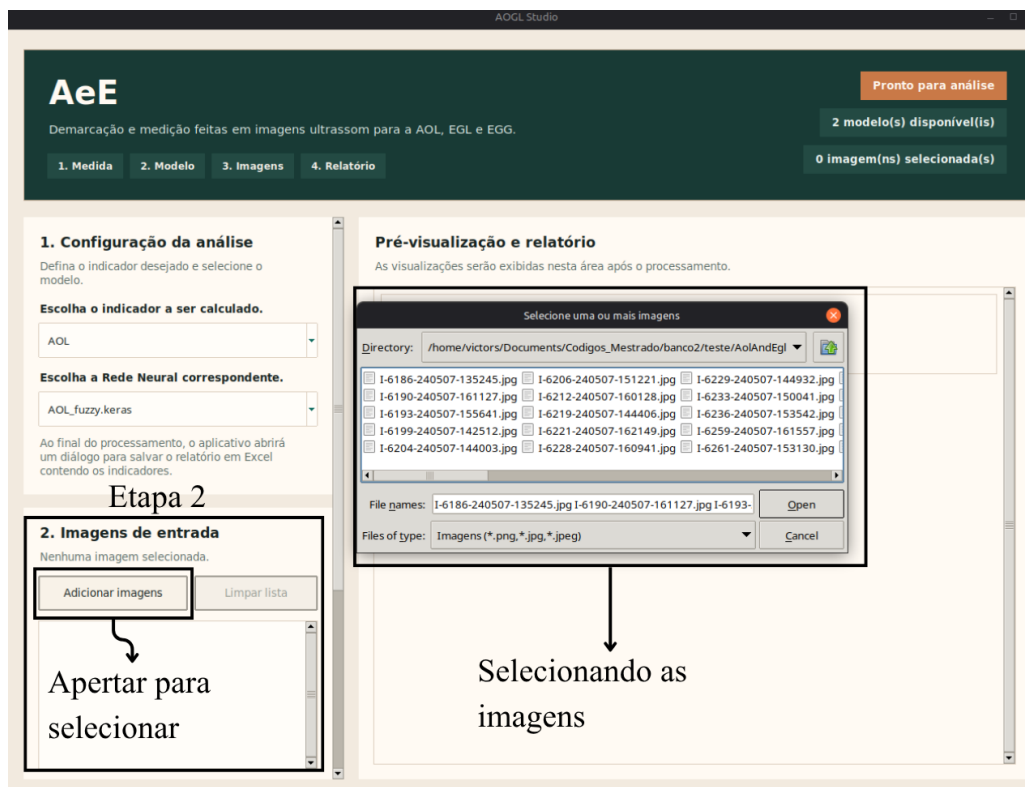


Fonte: Elaborado pelo autor.

- A segunda etapa é responsável pela seleção da(s) imagem(ns) de ultrassom a ser(em) processada(s). O usuário pode selecionar uma ou mais imagens, e o aplicativo irá processar cada uma delas utilizando o modelo selecionado na etapa anterior. Nesta etapa é

necessário que as imagens tenham a informação da raça do animal, neste caso, imagens que começam com “6” são da raça Nelore e imagens que começam com “2” são da raça Caracu. Na Figura 4.19 é mostrado a tela de seleção das imagens, em que são selecionadas três imagens de ultrassom para serem processadas.

Figura 4.19: Tela de seleção das imagens de ultrassom a serem processadas.



Fonte: Elaborado pelo autor.

Caso o indicador seja o AOL, todas as imagens selecionadas precisam ser do músculo *longuissimos dorsi*, para que seja possível medir corretamente. A imagem selecionada não precisa passar por nenhum processo de pré-processamento, ou seja, o aplicativo é capaz de processar a imagem original, sem a necessidade de redimensionamento ou normalização prévia. O aplicativo realiza automaticamente o pré-processamento necessário para que a imagem seja compatível com os modelos de segmentação.

- A terceira etapa é a visualização dos resultados. Após o processamento das imagens, o aplicativo exibe a máscara de segmentação gerada para cada imagem, sobreposta à imagem original, e apresenta os valores numéricos dos indicadores correspondentes, gerando também um relatório em formato de Excel, contendo o número do animal e a medida do indicador. Na Figura 4.20 é mostrado a tela de visualização dos resultados, em que são exibidas as máscaras de segmentação para as três imagens selecionadas e os valores numéricos do indicador AOL.

Figura 4.20: Tela de visualização dos resultados, com as máscaras de segmentação sobrepostas à imagem original e os valores numéricos do indicador.

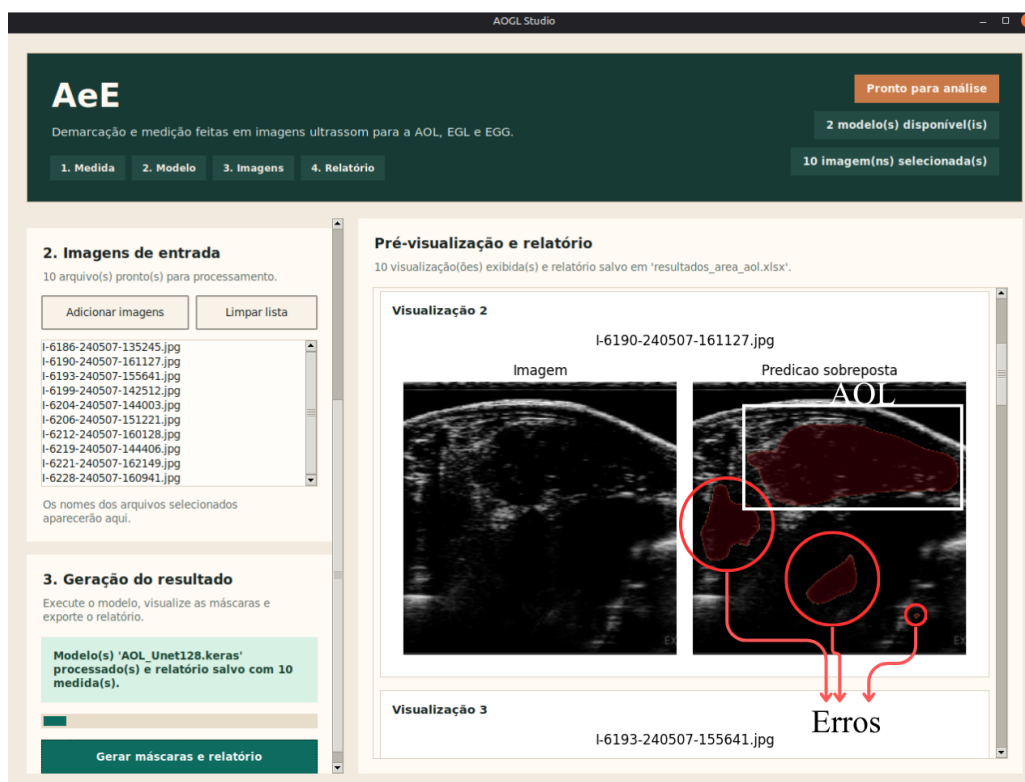


Fonte: Elaborado pelo autor.

Antes que se possa visualizar os resultados, é necessário salvar o relatório gerado, com o aplicativo solicitando o local onde o arquivo Excel deve ser salvo.

O aplicativo contém algumas funções que reduzem o erro da rede, como por exemplo, para a AOL, caso a imagem de entrada seja ruim e a rede “alucine”, gerando regiões que não correspondem à região de interesse, o aplicativo contém funções que eliminam regiões menores que um determinado número de pixels, o que ajuda a reduzir o impacto de segmentações errôneas. Na Figura 4.21 é mostrado um exemplo de uma imagem de ultrassom com baixa qualidade, em que a rede gerou uma segmentação errônea, e o aplicativo calculou apenas a região maior, eliminando as regiões menores.

Figura 4.21: Exemplo de uma imagem de ultrassom com baixa qualidade, em que a rede gerou uma segmentação errônea, e o aplicativo calculou apenas a região maior, eliminando as regiões menores.



Fonte: Elaborado pelo autor.

Neste capítulo foi abordado o contexto do problema, a manipulação feita no banco de dados, os resultados das redes neurais utilizadas e o desenvolvimento do aplicativo. No próximo capítulo, são apresentadas as considerações finais do trabalho, incluindo as principais conclusões, limitações e sugestões para trabalhos futuros.

# Capítulo 5

## Considerações Finais

Os resultados obtidos neste trabalho para as redes neurais desenvolvidas são satisfatórios para ambos os modelos. Os objetivos gerais e específicos, descritos na introdução, foram alcançados. Inicialmente, buscou-se compreender os conceitos fundamentais de redes neurais, teoria dos conjuntos fuzzy e sistemas baseados em regras fuzzy, possibilitando a junção da arquitetura U-Net com o ANFIS. A implementação do modelo U-Net Fuzzy possibilitou um amplo arcabouço teórico e prático. O desenvolvimento do aplicativo também foi um passo importante, tornando acessível a forma que se obtêm as medidas numéricas a partir das imagens de ultrassom, o que pode ser útil para profissionais da área de zootecnia e veterinária, especialmente, para o Centro Avançado de Pesquisa e Desenvolvimento de Bovinos de Corte.

Como citado ao decorrer da dissertação, um trabalho semelhante foi feito por Melo *et al.* (2022), entretanto, apenas para o indicador de AOL da raça Nelore, obtendo uma acurácia de 97,53% para a U-Net. Em contrapartida, neste trabalho obteve-se os seguintes resultados:

- O desenvolvimento de dois modelos de redes neurais multitarefas que além de segmentar as regiões de interesse, classificam a imagem quanto à sua raça (para este trabalho, Nelore ou Caracu);
- O uso de uma abordagem multitarefa nas arquiteturas desenvolvidas permitem a adição de novas raças bovinas para serem analisadas;
- Para o indicador AOL, o modelo U-Net Fuzzy obteve acurácia, coeficiente de Dice e MAE de 95,15%, 93,29% e 2,312 cm<sup>2</sup>, respectivamente, enquanto a U-Net Multitarefa obteve acurácia de 94,75%, coeficiente de Dice de 92,73% e MAE de 2,988 cm<sup>2</sup>;
- Para o indicador EGL, o modelo U-Net Fuzzy e U-Net Multitarefa obtiveram acurácia, coeficiente de Dice e MAE de 99,95% e 99,85%, 68,28% e 68,10% e 0,015 mm e 0,018 mm, respectivamente;
- Para o indicador EGG, o modelo U-Net Fuzzy obteve uma acurácia de 98,95% para o *Biceps Femoris* e 96,88% para a região de gordura e um coeficiente de Dice de 84,60% para o *Biceps Femoris* e 79,99% para a região de gordura, com MAE de 0,8 mm. A U-Net Multitarefa obteve acurácia de 98,75% para o *Biceps Femoris* e 96,87% para a

região de gordura, um coeficiente de Dice de 82,74% para o *Biceps Femoris* e 79,84% para a região de gordura, com MAE de 0,7 mm;

- O desenvolvimento de um aplicativo computacional capaz de detectar regiões específicas e medir automaticamente os indicadores de interesse em imagens de ultrassom de bovinos utilizando ambos os modelos, podendo auxiliar nos processos de seleção e manejo dos rebanhos.

O resultado superior da acurácia da AOL em Melo *et al.* (2022) pode estar associado à qualidade das imagens e dos *ground truth* utilizados, sendo estas, as principais dificuldades encontradas durante o desenvolvimento deste trabalho.

Para que uma imagem seja considerada de boa qualidade, é necessário seguir um protocolo de aquisição e análise rigoroso, como explicado por Mercadante *et al.* (2010). Cada imagem de ultrassom do músculo *longissimus dorsi* deve conter em posições específicas 19 pontos anatômicos bem definidos, o que não foi possível neste trabalho devido à quantidade limitada de imagens disponíveis. Por conta disso, os modelos treinados podem propagar erros que seriam facilmente evitáveis com imagens de melhor qualidade. A presença de artefatos, ruídos e variações na qualidade das imagens pode comprometer a precisão dos modelos de segmentação e, consequentemente, a precisão das medidas numéricas derivadas das máscaras de segmentação.

Outro desafio importante foi o desenvolvimento da U-Net Fuzzy. Como não foram encontradas, na literatura, implementações de uma arquitetura U-Net integrada ao ANFIS, foi necessário estudar e testar diferentes formas de realizar a integração. Esse processo exigiu diversas adaptações e experimentações, tornando o desenvolvimento complexo, mas também motivador.

Além disso, o módulo fuzzy foi implementado sem o uso de bibliotecas especializadas para a implementação do ANFIS. Embora essa abordagem tenha proporcionado maior controle sobre o funcionamento do sistema, também trouxe desafios relacionados à otimização do modelo, tendo em vista que, considerado um cenário com 3 funções de pertinência e 4 antecedentes, têm-se  $3^4 = 81$  regras fuzzy. Cada uma dessas regras possui uma matriz de pesos, que para o caso da arquitetura desenvolvida, tem dimensões  $1 \times 256$  e um vetor de viés de dimensões  $256 \times 1$ . Como consequência, apenas o módulo fuzzy passa a ter um total de  $81 \times 256 + 81 \times 256 \times 1 = 41.472$  parâmetros treináveis. Esse número é expressivo quando comparado ao próprio módulo multi-tarefa utilizada como base, que possui 512 parâmetros. Ainda assim, as redes desenvolvidas são relativamente pequenas, essa característica representa uma vantagem em termos de eficiência computacional, pois favorece a implementação em ambientes com recursos limitados e abre a possibilidade de adaptação do aplicativo desenvolvido para dispositivos móveis.

É importante destacar que os modelos desenvolvidos apresentam certa dependência em relação ao aparelho de ultrassom utilizado para a aquisição das imagens. Esta dependência ocorre já que o formato da imagem pode variar significativamente entre diferentes equipamentos, o que pode limitar a generalização dos resultados para outras situações. Uma possível solução para esse problema seria a utilização de técnicas que retirem informações específicas do aparelho, deixando apenas as informações relevantes para a segmentação.

O aplicativo desenvolvido também apresenta uma dependência relacionada ao formato dos dados de entrada. Atualmente, a identificação da raça do animal é realizada a partir do nome do arquivo da imagem. Mais especificamente, o primeiro dígito do nome é utilizado para determinar a raça: quando esse dígito é “6”, o animal é classificado como Nelore; quando é “2”, como Caracu. Embora essa abordagem seja adequada para o conjunto de dados utilizado neste trabalho, pode não ser aplicável a imagens provenientes de outras fontes ou equipamentos. Dessa forma, para ampliar a aplicabilidade do aplicativo. Uma alternativa seria exigir que o usuário informasse previamente a raça dos animais analisados ou utilizasse apenas imagens de uma raça específica. Outra possibilidade seria empregar técnicas de visão computacional capazes de identificar automaticamente a raça do animal a partir da própria imagem, tornando o sistema mais flexível e independente da convenção adotada na nomenclatura dos arquivos.

Como trabalhos futuros, podem ser estudados e implementados modelos que utilizem o algoritmo de atenção, desenvolvido pela Google (VASWANI *et al.*, 2017), em conjunto com a U-Net, como também a utilização de técnicas de transferência de aprendizado (*fine tuning*), que podem ajudar a melhorar o desempenho dos modelos, especialmente quando a quantidade e qualidade de dados disponíveis é limitada. O banco de dados também poderia passar por um filtro mais rigoroso de qualidade, para garantir que as imagens utilizadas para treinamento e avaliação dos modelos sejam de alta qualidade, seguindo os protocolos recomendados para a aquisição de imagens de ultrassom de bovinos, como também um aumento do número de imagens disponíveis, o que pode contribuir para a melhoria do desempenho dos modelos. Por fim, o aplicativo desenvolvido poderia ser adaptado para dispositivos móveis, tornando-o mais acessível e prático para os profissionais da área de zootecnia e veterinária.

# Referências Bibliográficas

- ABCCARACU. *História da raça caracu*. 2022. Acesso em 2026-04-03. Disponível em: <https://www.abccaracu.com.br/historia-da-raca-caracu>.
- ABIEC. *Beef Report — Perfil da Pecuária no Brasil*. 2021. Acesso em 2026-04-3. Disponível em: <https://abiec.com.br/publicacoes/beef-report-2025-perfil-da-pecuaria-no-brasil/>.
- Aloka Co. Ltd. *Aloka Echo Camera Model SSD-500: Operator's Manual*. Japan, 1990. Manual No. MN1-0346, Effective software version 5.0, issued August 27, 1990.
- AMIRI, M.; BROOKS, R.; BEHBOODI, B.; RIVAZ, H. Two-stage ultrasound image segmentation using U-Net and test time augmentation. *International Journal of Computer Assisted Radiology and Surgery*, v. 15, n. 6, p. 981–988, 2020.
- BEAR, M. F.; CONNORS, B. W.; PARADISO, M. A. *Neurociências: desvendando o sistema nervoso*. 4. ed. Porto Alegre: Artmed, 2017. ISBN 9788582713091.
- BENGIO, Y. Practical recommendations for gradient-based training of deep architectures. *Cornell University*, Version 2, 2012.
- BURDEN, R. *Numerical Analysis*. 3. ed. Boston, MA, USA: Cengage Learning, 2016. ISBN 978-8522123407.
- CHOWDHURY, M. *Padding and Strides in CNN*. 2024. Acesso em 2026-04-14. Disponível em: <https://medium.com/@minhazc.engg/padding-and-strides-in-cnn-58dc56493887>.
- DATA SCIENCE ACADEMY. *Comparativo Técnico – Label Encoding vs One-Hot Encoding em Machine Learning*. 2025. Acesso em 2026-04-13. Disponível em: <https://blog.dsacademy.com.br/comparativo-tecnico-label-encoding-vs-one-hot-encoding-em-machine-learning/>.
- DEEP LEARNING BOOK. *Capítulo 10: As Principais Arquiteturas de Redes Neurais*. 2025. Acesso em 2026-03-26. Disponível em: <https://www.deeplearningbook.com.br/>.
- DEEP LEARNING BOOK. *Capítulo 19: Overfitting e Regularização: Parte 1*. 2025. Acesso em 2026-01-26. Disponível em: <https://www.deeplearningbook.com.br/overfitting-e-regularizacao-parte-1/>.
- DEEP LEARNING BOOK. *Capítulo 22 – Regularização L1*. 2025. Acesso em 2026-03-26. Disponível em: <https://www.deeplearningbook.com.br/>.
- DICE, L. R. Measures of the amount of ecologic association between species. *Ecology*, v. 26, n. 3, p. 297–302, 1945.
- ELHARROUSS, O.; MAHMOOD, Y.; BECHQITO, Y.; SERHANI, M. A.; BADIDI, E.; RIFFI, J.; TAIRI, H. Loss functions in deep learning: A comprehensive review. *arXiv preprint arXiv:2504.04242*, 2025. Disponível em: <https://arxiv.org/abs/2504.04242>.

- FELICIO, P. E. Classificação, tipificação e qualidade da carne bovina. *VI Congresso Brasileiro de Ciência e Tecnologia de Carnes*, São Pedro, SP, Brasil, p. 9, 2010.
- GARCAO, L. M. C. Características socioeconômicas da pecuária brasileira: características socioeconômicas de los bovinos brasileños. *Revista Mirante*, Anápolis, GO, Brasil, jun. 2015. ISSN 1981-4089.
- GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016. <http://www.deeplearningbook.org>.
- GOOGLE LLC. *TensorFlow*. 2015. Acesso em: 11 abr. 2026. Disponível em: <https://www.tensorflow.org/>.
- GUARALDO, M. C. *Brasil é o quarto maior produtor de grãos e o maior exportador de carne bovina do mundo, diz estudo*. 2021. Acesso em 2026-04-3. Disponível em: <https://www.embrapa.br/busca-de-noticias/-/noticia/62619259/>.
- GUO, Y.; DUAN, X.; WANG, C.; GUO, H. Segmentation and recognition of breast ultrasound images based on an expanded U-Net. *PLOS ONE*, v. 16, n. 6, p. e0253202, 2021.
- HECHT-NIELSEN, R. Theory of the backpropagation neural network. *Academic Press*, 1992.
- HUMANSIGNAL. *Label Studio: Open Source Data Labeling Platform*. 2020. Acesso em: 11 abr. 2026. Disponível em: <https://labelstud.io/>.
- JAFELICE, R. S. M.; BARROS, L. C.; BASSANEZI, R. C. *Teoria dos Conjuntos Fuzzy com Aplicações*. 3. ed. São Paulo -SP, Brasil: SBMAC, 2023. ISBN 978-65-86399-14-5.
- JANG, J.-S. R. ANFIS: Adaptive-network-based fuzzy inference system. *IEEE Transactions on Systems, Man, and Cybernetics*, v. 23, n. 3, p. 665–685, 1993.
- KERAS TEAM. *Keras: Deep Learning for Humans*. 2015. Acesso em: 11 abr. 2026. Disponível em: <https://keras.io/>.
- KIRICHEV, M. M.; SLAVOV, T. S.; MOMCHEVA, G. D. Fuzzy U-net neural network architecture optimization for image segmentation. *IOP Conference Series: Materials Science and Engineering*, IOP Publishing, 2021.
- LECUN, Y. *The MNIST Database of Handwritten Digits*. 1998. Acesso em 2026-02-22. Disponível em: <http://yann.lecun.com/exdb/mnist/>.
- LETTVIN, J. Y.; MATURANA, H. R.; MCCULLOCH, W. S.; PITTS, W. H. What the frog's eye tells the frog's brain. *Proceedings of the Institute of Radio Engineers*, v. 47, n. 11, p. 1940–1951, 1959.
- LINUX MINT TEAM. *Linux Mint*. 2026. Acesso em: 23 abr. 2026. Disponível em: <https://linuxmint.com/>.
- MALEKMOHAMMADI, A.; SORYANI, M.; KOZEGAR, E. Mass segmentation in automated breast ultrasound using an enhanced attentive u-net. *Expert Systems with Applications*, v. 245, p. 123095, 2024.
- MCCULLOCH, W.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 1943.

- MELO, M. J.; GONÇALVES, D. N.; GOMES, M. N. B.; FARIA, G.; SILVA, J. A.; RAMOS, A. P. M.; OSCO, L. P.; FURUYA, M. T. G.; JUNIOR, J. M.; GONÇALVES, W. N. Automatic segmentation of cattle rib-eye area in ultrasound images using the UNet++ deep neural network. *Computers and Electronics in Agriculture*, Elsevier, v. 195, p. 106818, 2022.
- MERCADANTE, M. E. Z.; SILVA, S. da L.; BUENO, M. S.; TAROUÇO, J. U.; YOKOO, M. J. I. Repetibilidade da mensuração de imagens das características de carcaça obtidas por ultrassonografia em fêmeas nelore. *Revista Brasileira de Zootecnia*, v. 39, n. 4, p. 752–757, 2010. ISSN 1806-9290.
- MICROSOFT. *Visual Studio Code*. 2015. Acesso em: 23 abr. 2026. Disponível em: <https://code.visualstudio.com/>.
- MÜLLER, V. C. Review of margaret boden, “mind as machine: A history of cognitive science”. *Minds and Machines*, Springer, v. 18, n. 1, p. 121–125, 2008.
- O’SHEA, K.; NASH, R. An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*, 2015. Disponível em: <https://arxiv.org/abs/1511.08458>.
- PRAMANIK, P.; PRAMANIK, R.; SCHWENKER, F.; SARKAR, R. DBU-Net: Dual branch U-Net for tumor segmentation in breast ultrasound images. *PLOS ONE*, v. 18, n. 11, p. e0293615, 2023.
- PURWONO, P.; NATALIANI, Y.; PURNOMO, H. D.; TIMOTIUS, I. K. Dfu-fuzzyliteunet: A lightweight U-net with fuzzy sigmoid and lite transformer for diabetic foot ulcer segmentation. *Biomedical Signal Processing and Control*, Elsevier, v. 108, p. 107902, 2025.
- PYTHON SOFTWARE FOUNDATION. *Python*. 1990. Acesso em: 23 abr. 2026. Disponível em: <https://www.python.org/>.
- PYTHON SOFTWARE FOUNDATION. *tkinter — Python interface to Tcl/Tk*. 1991. Acesso em: 23 abr. 2026. Disponível em: <https://docs.python.org/3/library/tkinter.html>.
- RONNEBERGER, O.; FISCHER, P.; BROX, T. U-net: Convolutional networks for biomedical image segmentation. In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. Munich, Germany: Springer, 2015. p. 234–241.
- ROSENBLATT, F. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 1958.
- RUDER, S. An overview of gradient descent optimization algorithms. *Insight Centre for Data Analytics, NUI Galway Aylien Ltd., Dublin*, 2016.
- SENA, V.; JAFELICE, R. S. M.; SANTOS, C. A. R. Reconstrução geométrica de uma função através de uma rede neural. In: *Anais da XXIV Semana da Matemática e XIV Semana da Estatística (SEMAT & SEMEST)*. Uberlândia, MG, Brasil: UFU, 2024. Disponível em: [https://drive.google.com/file/d/1OmN2Gq6i3nHR7ZtEVitc5iCVH0ZFUY\\_8/view](https://drive.google.com/file/d/1OmN2Gq6i3nHR7ZtEVitc5iCVH0ZFUY_8/view).
- SENA, V.; JAFELICE, R. S. M.; SANTOS, C. A. R. Reconstrução geométrica de uma função através de uma rede neural. In: *Caderno de Resumos do I Workshop de Matemática da UFU*. Uberlândia, MG, Brasil: UFU, 2025. Disponível em: <https://eventos.ufu.br/ime/wm>.
- SOVIERZOSKI, M. A. Convolução de sinais: definição, propriedades e ferramentas. *Revista Ilha Digital*, v. 2, p. 81–95, 2010. ISSN 2177-2649. Disponível em: <https://ilhadigital.florianopolis.ifsc.edu.br/index.php/ilhadigital>.

STANFIELD, C. L. *Fisiologia Humana*. São Paulo: Pearson Education do Brasil, 2014. ISBN 978-85-430-1426-5.

SUGUISAWA, L.; MATOS., B. C.; SUGUISAWA, J. M. Uso da ultrassonografia na avaliação de características de carcaça e de qualidade da carne. In: *Melhoramento Genético Aplicado em Gado de Corte*. Brasília, DF, Brasil: Embrapa, 2013. p. 98–107.

SUTSKEVER, I. On the importance of initialization and momentum in deep learning. *International Conference on Machine Learning*, 2013.

SZELISKI, R. *Computer Vision: Algorithms and Applications*. Cham, Suíça: Springer, 2010. ISBN 978-3030343712.

SZENTIMREY, Z.; RIBAUPIERRE, S. de; FENSTER, A.; UKWATTA, E. Automated 3D U-net based segmentation of neonatal cerebral ventricles from 3D ultrasound images. *Medical Physics*, v. 49, n. 2, p. 1034–1046, 2022.

TileStats. *U-Net Clearly Explained — Image Segmentation with AI*. 2025. Vídeo publicado em 23 jun. 2025. Acesso em: 05 jun. 2026. Disponível em: <https://www.youtube.com/watch?v=oxcgv75k6yU&t=1324s>.

TREVISAN, V. *Como funcionam as Redes Neurais Convolucionais (CNNs)*. 2021. Acesso em 2025-12-16. Disponível em: <https://medium.com/data-hackers/como-funcionam-as-redes-neurais-convolucionais-cnns-71978185c1>.

UNITED STATES DEPARTMENT OF AGRICULTURE. *Livestock and Poultry: World Markets and Trade*. Washington, DC, USA, 2026. Acesso em: 23 abr. 2026. Disponível em: <https://www.fas.usda.gov/data/livestock-and-poultry-world-markets-and-trade>.

VASWANI, A.; SHAZEER, N.; PARMAR, N.; USZKOREIT, J.; JONES, L.; GOMEZ, A. N.; KAISER, L.; POLOSUKHIN, I. Attention is all you need. In: *Advances in Neural Information Processing Systems 30 (NeurIPS 2017)*. Curran Associates, Inc., 2017. p. 5998–6008. Disponível em: <https://arxiv.org/abs/1706.03762>.

VERMA, H.; IM, K.; GUPTA, A.; TANVEER, M. A novel framework using intuitionistic fuzzy logic with u-net and u-net++ architecture: A case study of mri brain image segmentation. *arXiv preprint arXiv:2603.18042*, 2026. Disponível em: <https://arxiv.org/abs/2603.18042>.

WERBOS, P. J. Backpropagation through time: What it does and how to do it? *PROCEEDINGS OF THE IEEE*, 1990.

WILSON, D. E. *Scanning into the Future: Real-Time Ultrasound in Beef Cattle*. 1998. ASB 1998-DEW-417. Disponível em: <https://admin.webplus.com.br/public/upload/downloads/090220121626346252000AHZV.pdf>.

ZADEH, L. A. Fuzzy sets. *Information and Control*, v. 8, n. 3, p. 338–353, 1965.

ZANCANER, R. *História da raça nelore*. 2026. Acesso em 2026-04-03. Disponível em: <https://zancaner.com.br/site/conteudo/1-historia-da-raca-nelore.html>.

ZHENG, T.; QIN, H.; CUI, Y.; WANG, R.; ZHAO, W.; ZHANG, S.; GENG, S.; ZHAO, L. Segmentation of thyroid glands and nodules in ultrasound images using the improved U-Net architecture. *BMC Medical Imaging*, v. 23, n. 1, p. 56, 2023.

ZHU, H.; ROHLING, R.; SALCUDEAN, S. Multi-task unet: Jointly boosting saliency prediction and disease classification on chest x-ray images. *arXiv preprint arXiv:2202.07118*, 2022. Disponível em: [⟨https://arxiv.org/abs/2202.07118⟩](https://arxiv.org/abs/2202.07118).

## ANEXO A - Comissão de Ética

**UNIVERSIDADE FEDERAL DE UBERLÂNDIA**

Reitoria

Pró-Reitoria de Pesquisa e Pós-Graduação

Comissão de Ética na Utilização de Animais

Rua Ceará, s/n, Bloco 2D, Sala 02 - Bairro Umarama, Uberlândia-MG, CEP 38400-902

Telefone: (34) 3225-8658 - [www.comissoes.propp.ufu.br/ceua](http://www.comissoes.propp.ufu.br/ceua) - [ceua@propp.ufu.br](mailto:ceua@propp.ufu.br)

**PARECER Nº** 74/2025/CEUA/PROPP/REITO  
**PROCESSO Nº** 23117.080958/2025-21  
**INTERESSADO(S):** AMANDA MARCHI MAIORANO

**ANÁLISE FINAL Nº 23117.080958/2025-21 DA COMISSÃO DE ÉTICA NA UTILIZAÇÃO DE ANIMAIS**

**Projeto Pesquisa:** " Integração de ultrassonografia e inteligência artificial para avaliação automatizada de características de carcaça em bovinos das raças Caracu e Nelore".

**Pesquisador Responsável:** Amanda Marchi Maiorano

Declaro para os devidos fins, que o projeto intitulado " Integração de ultrassonografia e inteligência artificial para avaliação automatizada de características de carcaça em bovinos das raças Caracu e Nelore", **envolve a avaliação de prontuários hospitalares ou banco de dados**. Desse modo, por não ferir a ética relacionada à experimentação animal e, estando em conformidade com a legislação federal, Lei Nº 11.794, de 8 de outubro de 2008, não há necessidade de apreciação e aprovação pela Comissão de Ética na Utilização de Animais-CEUA.

**Data Início:** 25/01/2026

**Data Término:** 31/08/2026

**LUIZ FERNANDO MOREIRA IZIDORO****Coordenador da Comissão de Ética na Utilização de Animais (CEUA)****Portaria R Nº 6794/2024**

Documento assinado eletronicamente por **Luiz Fernando Moreira Izidoro, Presidente**, em 08/12/2025, às 09:16, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do [Decreto nº 8.539, de 8 de outubro de 2015](#).



A autenticidade deste documento pode ser conferida no site [https://www.sei.ufu.br/sei/controlador\\_externo.php?acao=documento\\_conferir&id\\_orgao\\_acesso\\_externo=0](https://www.sei.ufu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0), informando o código verificador **6920801** e o código CRC **8F2CD355**.

---

---

**Referência:** Processo nº 23117.080958/2025-21

SEI nº 6920801