Universidade Federal de Juiz de Fora
Instituto de Ciências Exatas
Bacharelado em Ciência da Computação

Uma meta-heurística inspirada em morcegos para otimização da estrutura de redes neurais convolucionais para classificação de raças de bovinos

Daniel Muller Rezende

JUIZ DE FORA
AGOSTO, 2025

Uma meta-heurística inspirada em morcegos para otimização da estrutura de redes neurais convolucionais para classificação de raças de bovinos

Daniel Muller Rezende

Universidade Federal de Juiz de Fora

Instituto de Ciências Exatas

Departamento de Ciência da Computação

Bacharelado em Ciência da Computação

Orientador: Saulo Moraes Villela

Coorientador: Luiz Maurílio da Silva Maciel

JUIZ DE FORA

AGOSTO, 2025

Uma meta-heurística inspirada em morcegos para otimização da estrutura de redes neurais convolucionais para classificação de raças de bovinos

Daniel Muller Rezende

MONOGRAFIA SUBMETIDA AO CORPO DOCENTE DO INSTITUTO DE CIÊNCIAS EXATAS DA UNIVERSIDADE FEDERAL DE JUIZ DE FORA, COMO PARTE INTEGRANTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE BACHAREL EM CIÊNCIA DA COMPUTAÇÃO.

Aprovada por:

Saulo Moraes Villela
Doutor em Engenharia de Sistemas e Computação

Luiz Maurílio da Silva Maciel
Doutor em Engenharia de Sistemas e Computação

Marcelo Bernardes Vieira
Doutor em Ciência da Computação

Heder Soares Bernardino
Doutor em Modelagem Computacional

JUIZ DE FORA
13 DE AGOSTO, 2025

Este trabalho é dedicado aos meus pais e ao meu irmão, por todo apoio e incentivo aos meus estudos, carinho e dedicação em cada etapa da minha caminhada.

Resumo

A pecuária é uma das atividades econômicas mais relevantes globalmente, e o bem-estar animal tem ganhado destaque, impulsionando o uso de tecnologias automatizadas para aumentar a eficiência e a sustentabilidade do setor. Nesse contexto, a Visão Computacional, uma área de pesquisa voltada para a extração de informações a partir de imagens e vídeos, tem se mostrado promissora. O estado da arte nessa área é dominado por métodos de aprendizado profundo, especialmente com uso das redes neurais convolucionais, amplamente aplicadas em tarefas de classificação, detecção e reconhecimento de padrões visuais. Este trabalho propõe uma abordagem para modificar a estrutura interna de redes profundas, com foco na profundidade e largura das redes, utilizando uma metaheurística bioinspirada no comportamento de uma população de morcegos. A proposta visa ajustar arquiteturas da literatura para um problema específico de classificação de raças bovinas, contribuindo para maior eficiência computacional e melhor desempenho. A partir de dois modelos bem estabelecidos (VGGNet e DenseNet), o algoritmo foi capaz de gerar versões otimizadas com desempenho superior ou aproximado ao dos modelos originais. A VGGNet ajustada apresentou o melhor resultado, alcançando maior acurácia média nos conjuntos de validação e teste, além de uma expressiva redução no custo computacional, por exemplo. Os testes realizados evidenciam que o algoritmo proposto é eficaz na criação de arquiteturas mais leves e precisas, adaptadas ao problema de interesse. Apesar dos avanços, o método ainda apresenta limitações, como o elevado tempo de execução decorrente do número de treinos exigidos, e a sensibilidade estrutural das redes utilizadas. Ainda assim, a abordagem mostra potencial de expansão para outras tarefas de Visão Computacional, como detecção e segmentação de objetos.

Palavras-chave: Classificação de imagens. Redes Neurais. Otimização. Algoritmos bioinspirados. Pecuária de Precisão.

Abstract

Livestock farming is one of the most important economic activities globally, and animal welfare has gained prominence, driving the use of automated technologies to increase the sector's efficiency and sustainability. In this context, Computer Vision, a research area focused on extracting information from images and videos, has shown promise. The state-of-the-art in this field is dominated by deep learning methods, especially those using convolutional neural networks, widely applied in visual pattern classification, detection, and recognition tasks. This work proposes an approach to modify the internal structure of deep networks, focusing on their depth and width, using a metaheuristic bioinspired by the behavior of a bat population. The proposal aims to adjust architectures from the literature for a specific cattle breed classification problem, contributing to greater computational efficiency and improved performance. Using two well-established models (VGGNet and DenseNet), the algorithm was able to generate optimized versions with performance superior to or close to that of the original models. The adjusted VGGNet presented the best result, achieving higher average accuracy in the validation and test sets, in addition to a significant reduction in computational cost, for example. The tests performed demonstrate that the proposed algorithm is effective in creating lighter and more accurate architectures, adapted to the problem of interest. Despite the advances, the method still has limitations, such as the high execution time due to the number of training sessions required, and the structural sensitivity of the networks used. Even so, the approach shows potential for expansion to other Computer Vision tasks, such as object detection and segmentation.

Keywords: Image Classification. Neural Networks. Optimization. Bio-inspired Algorithms. Precision Livestock Farming.

Agradecimentos

Agradeço aos meus queridos pais, Pierre e Telma, e ao meu irmão Caio, por estarem comigo me incentivando e apoiando em toda a minha caminhada na vida acadêmica e profissional. Agradeço imensamente todos os ensinamentos, carinho e por serem meus melhores amigos em cada etapa de minha vida. Se eu sou quem eu me tornei hoje e cheguei aonde estou, foi por causa de vocês. Cada risada, cada atenção, cada momento feliz ou triste ao lado de vocês, foram assenciais para meu amadurecimento e evolução.

À minha bela e amada Isa, que tornou minha vida muito mais alegre e bonita. Sua presença, apoio e companheirismo me inspiram a continuar lutando e ser cada dia melhor. Seu amor e carinho incondicionais foram essenciais para que eu superasse meus obstáculos e alcançasse meus objetivos. Sou o mais feliz por ter você em minha vida.

Ao meu amigo Pedro Marangom, agradeço por todos os anos de amizade que compartilhamos juntos. Apesar de nossos caminhos terem se distanciado após o ensino médio, cada momento que nos encontramos, mesmo que não tão frequente como gostaríamos, mostra como nossa amizade é forte. Sou muito grato por ser seu amigo.

Ao meu orientador Saulo Moraes, sou grato por todas as oportunidades que tive durante minha graduação, por todo o aprendizado e por estar comigo desde o meu primeiro projeto de pesquisa na Universidade Federal de Juiz de Fora. Agradeço também aos professores do projeto *Happy Cow ID*, Luiz Maurílio e Marcelo Bernardes, por todas as orientações dentro do projeto.

À Embrapa Gado de Leite pela oportunidade de participar do projeto *Happy Cow ID* e ao RePesq da UFJF pelo fornecimento dos recursos computacionais necessários para a realização dos experimentos presentes neste trabalho.

"A esperança é como o Sol... Se você só acredita quando o vê, nunca sobreviverá à noite".

Princesa Leia (Star Wars)

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Abreviações

CNN	*Convolutional Neural Network*
FC	*Fully Connected*
FGVC	*Fine-Grained Visual Categorization*
HPO	*Hyper Parameters Optimization*
IA	Inteligência Artificial
NP	*Non Deterministic Polynomial*
RGB	*Red Green Blue*
SGD	*Stochastic Gradient Descendent*

1 Introdução

A pecuária é uma atividade de grande importância na economia mundial e o Brasil figura como um importante fornecedor mundial de *commodities* agrícolas, com imenso potencial de produção e exportação. De acordo com Garbaccio e Pratlong (2024), o Brasil é um dos maiores produtores de carne do mundo, sendo responsável por fornecer 26% do consumo mundial de carne bovina e 12% de carne suína. Neste setor, a otimização dos processos nas fazendas é uma tendência global impulsionada pela necessidade de equilibrar a carga de trabalho com margens de lucro (SILVA et al., 2024), o que tem motivado, ao longo dos últimos anos, a incorporação de novas tecnologias na produção.

Diante dos desafios da pecuária moderna, a adoção de tecnologias de precisão tem se tornado cada vez mais comum, as quais utilizam sistemas de monitoramento por sensores que coletam dados individuais sobre os animais e o ambiente de forma automatizada. Esses sistemas geram grandes volumes de dados que são interpretados por softwares e modelos matemáticos, incluindo inteligência artificial. Através do uso dessas tecnologias, é possível garantir não apenas a qualidade e segurança da produção, como também a manutenção do bem-estar animal, através do monitoramento de indicadores produtivos e comportamentais, facilitando a identificação precoce de doenças e sinais clínicos (FERREIRA; SIQUEIRA; PEREIRA, 2015). Dessa forma, a busca por métodos que sejam eficazes em solucionar as principais demandas da pecuária se tornou um importante campo de pesquisa na área da tecnologia e ciência da computação.

A Visão Computacional é uma área de pesquisa na qual, a partir de uma imagem de entrada, o objetivo principal é obter como saída informações sobre a cena que deu origem a essa imagem (ROSENFELD, 1988). De acordo com Szeliski (2022), é possível dividir o problema de reconhecimento em um conjunto de tarefas. As principais envolvem a classificação, focada em categorizar imagens em conjuntos fixos de classes; a detecção de objetos, cujo objetivo é localizar instâncias de objetos em uma imagem; e a segmentação, que é semelhante à classificação, porém busca rotular os *pixels* da imagem individualmente. As tarefas de reconhecimento são a base para a solução de diversos problemas

de Visão Computacional, em diferentes setores da sociedade. Martinez, Al-Hussein e Ahmad (2019) mostram como técnicas de Visão Computacional podem ser utilizadas para garantir a segurança na indústria da construção. Outro exemplo de aplicação é abordado por Khemasuwan, Sorensen e Colt (2020), onde os autores discutem o uso de Inteligência Artificial (IA) e Visão Computacional na medicina pulmonar, impulsionada pelo crescente número de dados gerados por fontes digitais.

Dentre os principais métodos utilizadas atualmente, é possível citar o aprendizado profundo (*deep learning*), o qual consiste em simular o comportamento do cérebro humano através da representação hierárquica de alto nível dos dados de entrada (BEZERRA, 2016). Para isso, esses métodos utilizam camadas sequenciais em uma rede neural artificial, construída a partir do conceito de neurônio artificial proposto por Rosenblatt (1958). Esses mecanismos têm sido amplamente utilizados para a resolução de problemas e demandas da pecuária. Dac et al. (2022) por exemplo, utilizam aprendizado profundo para desenvolver um sistema de reconhecimento facial de vacas leiteiras, o qual, a partir de análise de vídeo, realiza a detecção, recorte, codificação e pesquisa facial do animal. Outro caso de aplicação pode ser visto em Wu et al. (2023), no qual os autores propõem um método para monitorar o comportamento respiratório de múltiplas vacas leiteiras em ambientes de criação lotados e variáveis. O estudo, que foca no monitoramento durante o estado de repouso dos indivíduos, uma vez que representa de forma mais aproximada o estado fisiológico, obteve acurácia média geral acima de 93% em 60 vídeos testados.

1.1 Definição do Problema

Tendo isso em vista, é notável como a utilização de técnicas de aprendizado profundo podem beneficiar os processos envolvidos com a pecuária nas grandes fazendas, reduzindo a necessidade de trabalhos manuais e aumentando a qualidade da produção. Dessa forma, o presente trabalho propõe o desenvolvimento de um modelo de classificação de raças de bovinos, o qual visa distinguir a qual raça um indivíduo presente em uma imagem pertence, especificamente Girolando ou Holandês. Obter a identificação correta dessa característica pode contribuir para o monitoramento da saúde dos animais e controle da produção, uma

vez que cada raça apresenta particularidades fisiológicas que podem afetar o diagnóstico de doenças.

Escolher qual a arquitetura de rede neural adequada para o problema em foco é uma tarefa manual e repetitiva, tendo em vista que cada problema possui um nível de dificuldade e características distintas. Neste sentido, aplicação indiscriminada de modelos pré-estabelecidos pode implicar em resultados insatisfatórios ou custo computacional desnecessário, o que torna as tarefas apoiadas em Visão Computacional ainda mais desafiadoras. Portanto, este trabalho propõe o desenvolvimento de um método baseado em algoritmos bioinspirados para a criação de uma arquitetura personalizada para a tarefa de classificação abordada anteriormente. O método deve ser capaz de se ajustar às particularidades do problema de identificação de raças de bovinos e equilibrar os resultados com o consumo de recursos computacionais.

Para que isso seja possível, é necessário, inicialmente, construir e rotular uma base de dados contendo imagens de indivíduos pertencentes às raças descritas, selecionar as arquiteturas que serão utilizadas para solucionar o problema, bem como propor o método de otimização desses modelos. Logo, este trabalho apresenta como principais contribuições a construção de um modelo específico para a o problema, uma base de dados estruturada para a classificação de raças de bovinos, bem como um método de otimização capaz de ajustar as dimensões de um modelo de classificação em outros problemas do mundo real.

1.2 Objetivos

Este trabalho possui como objetivo principal a criação de um modelo otimizado por uma metaheurística bioinspirada em uma população de morcegos, baseado em arquiteturas já estabelecidas na literatura, a qual seja capaz de solucionar o problema de classificação de raças de bovinos. Para isso, apresenta como objetivos específicos:

- Construção de um conjunto de dados de imagens de bovinos rotuladas com suas raças;
- Escolha e estudo das arquiteturas a serem utilizadas nos experimentos;

- Escolha, estudo e modelagem do algoritmo de otimização para as arquiteturas selecionadas;
- Realização dos experimentos de otimização e treinamento dos modelos;
- Avaliação do desempenho com base na comparação dos modelos otimizados com as arquiteturas originais em relação aos resultados para o problema de classificação;

1.3 Organização do Texto

Além da introdução, o presente trabalho é constituído de outros cinco capítulos nomeados respectivamente como "Fundamentação Teórica", "Conjunto de Dados", "Abordagem Proposta", "Experimentos e Resultados" e "Considerações Finais". A fundamentação teórica, Capítulo 2 é responsável por fornecer os fundamentos técnicos para o completo entendimento do problema abordado e o método proposto. Ela engloba explicações sobre o problema de classificação de imagens, redes neurais convolucionais para tarefas de classificação e os principais conceitos e mecanismos para otimização de modelos de aprendizado profundo, como a otimização de hiper-parâmetros e sua estrutura interna. Aliado a fundamentação, os trabalhos relacionados, presentes no Capítulo 3, fornecerão uma revisão dos textos e estudos sobre os assuntos abordados.

O Capítulo 4 detalha as características do conjunto de dados utilizado nos experimentos desenvolvidos. Nele, são apresentadas informações quantitativas e qualitativas sobre as imagens dos bovinos, bem como o processo de pré-processamento e organização do conjunto de dados para treinamento. Já o Capítulo 5 descreve todo o método desenvolvido neste trabalho, incluindo a configuração das arquiteturas utilizadas e a modelagem do algoritmo de otimização.

Após isso, o Capítulo 6 descreve os experimentos desenvolvidos para validar o método proposto. Nele, são apresentados as configurações e parâmetros estabelecidos para os testes, bem como os principais resultados obtidos. Por fim, o Capítulo 7 apresenta as considerações finais sobre esse trabalho, detalhando as vantagens, limitações e possibilidades de trabalhos futuros.

2 Fundamentação Teórica

Neste capítulo são apresentados os principais conceitos teóricos, com o objetivo de construir a base necessária para a compreensão dos estudos e métodos desenvolvidos neste trabalho. Para isso, inicialmente, detalha-se o conceito de problemas de classificação de imagens, explorando os principais métodos modernos utilizados, com destaque para redes neurais convolucionais (*Convolutional Neural Networks* – CNNs). Em seguida, apresenta-se os principais conceitos sobre meta-heurísticas para solucionar problemas de otimização com destaque para um algoritmo bioinspirado em uma população de morcegos.

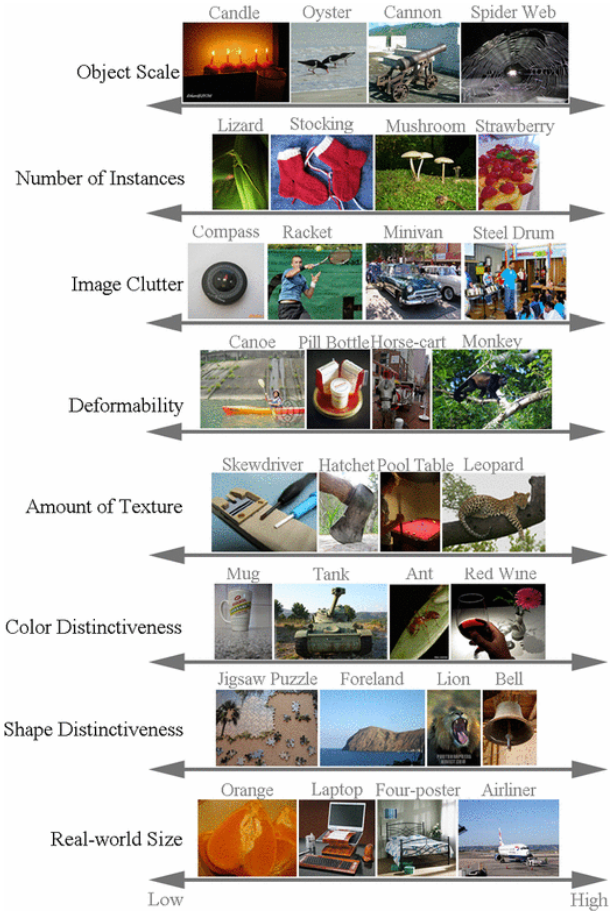
2.1 Classificação de Imagens

A classificação de imagens é uma das principais tarefas em Visão Computacional, tendo como objetivo categorizar uma imagem de entrada com base em suas características visuais (SINGH; SINGH, 2020). Essa categorização normalmente leva em consideração o objeto central e de maior destaque, fazendo com que a imagem inteira pertenca à classe desse objeto. De acordo com Ionescu et al. (2016), a dificuldade de uma imagem está relacionada ao tempo necessário para solucionar uma tarefa de busca visual. Neste sentido, essa dificuldade pode variar de acordo com a complexidade da cena, número de objetos, e desorganização do fundo (*background clutter*), o que torna a tarefa de classificação desafiadora, uma vez que informações extras presentes no fundo da imagem podem trazer ambiguidades entre as classes e acarretar no enviesamento dos resultados.

Para a criação de um modelo de classificação, é necessário a construção de uma base de dados estruturada com imagens do problema abordado e seus devidos rótulos. Diferentemente do problema de detecção, no qual é necessário anotar as coordenadas dos objetos de interesse (ADHIKARI; HUTTUNEN, 2021), o processo de rotulação dessa tarefa consiste apenas em definir a qual classe cada imagem pertence. Apesar de parecer simples, a definição de um conjunto de dados para classificação é um componente crítico, uma vez que a coleta de dados de treinamento representativos e imparciais em quantidades

suficientes é essencial para o sucesso dos modelos. Um exemplo de conjunto de dados para classificação de imagens é mostrado na Figura 1. Nela, é possível observar, além das classes, um conjunto de características dessas imagens que podem afetar a qualidade do conjunto de dados, como a escala, distinção de cores e tamanho dos objetos.

Figura 1: Exemplo de conjunto de dados para classificação de imagens.



Fonte: Russakovsky et al. (2015).

As dificuldades e desafios na classificação de imagens são diversos, especialmente quando é necessário lidar com subcategorias das classes gerais. Esse problema é denominado classificação visual fina (*Fine-Grained Visual Categorization* – FGVC) e consiste em lidar com a diferenciação de classes que possuem alta similaridade visual. Nele, os objetos, que normalmente pertencem a uma mesma classe geral, podem ser quase idênticos, diferindo uns dos outros por apenas sutis marcações. Gwilliam e Farrell (2020) mostram que a construção de conjunto de dados para este tipo de tarefa pode ser desafiadora, uma vez que a correta representação de todas as classes é crítica e não há disponibilidade igualitária de dados, especialmente daquelas que possuem uma alta dificuldade visual. Um

exemplo abordado por eles é a representação da fauna de regiões com alta biodiversidade e poucas observações, como o caso da Amazônia. A Figura 2 mostra um exemplo de classificação de subclasses em relação às respectivas classes gerais. A partir dessa visualização, deve-se definir a classificação de raças de bovinos como um problema que se encaixa nessa categoria.

Figura 2: Exemplo de conjunto de dados para classificação FGVC.

Fonte: Russakovsky et al. (2015).

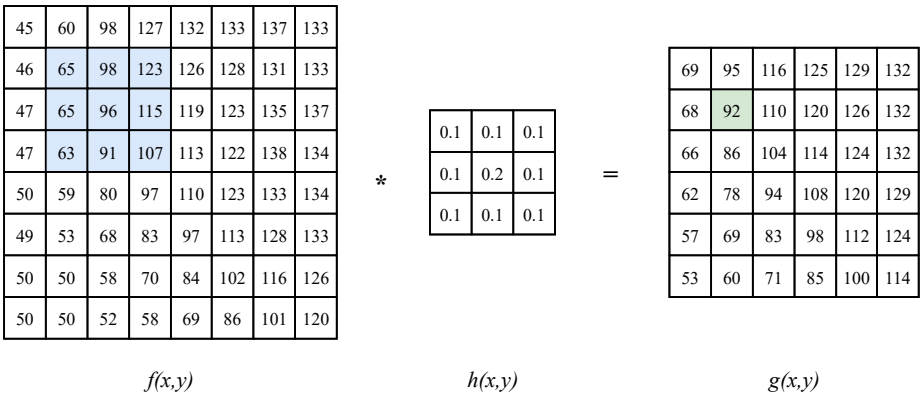
2.2 Redes Neurais para Classificação de Imagens

Ao longo dos anos, a necessidade de métodos aprimorados para a solução de problemas de Visão Computacional tornou o aprendizado profundo uma importante área de estudo e pesquisa, possibilitando o surgimento das Redes Neurais Convolucionais (*Convolutional Neural Network* – CNNs). O primeiro modelo, denominado LeNet, foi proposto por LeCun et al. (2002) para solucionar o problema de reconhecimento de dígitos, e possuía como particularidade em relação às redes profundas tradicionais, a capacidade de reconhecer informações em imagens. Diferentemente dos métodos tradicionais, os quais dependiam de configurações manuais, as CNNs são baseadas em uma abordagem de ponta a ponta (*end-to-end*), na qual todos os parâmetros do modelo são aprendidos durante o treinamento. Isso possibilita o aprendizado automático de informações provenientes das imagens.

As redes convolucionais são construídas seguindo os mesmos princípios de uma rede neural tradicional, porém com a particularidade de fazer uso de operações de con-

volução em múltiplas camadas (SZELISKI, 2022). Considerando uma imagem digital como uma matriz de *pixels* de duas dimensões, uma operação de convolução consiste em deslizar um *kernel* rotacionado em 180 graus sobre os *pixels* dessa imagem (GONZALEZ, 2009). A cada deslocamento é realizada uma soma de produtos entre os coeficientes do *kernel* e os *pixels* cobertos por esse *kernel*. Os coeficientes são responsáveis por definir a natureza do filtro a ser aplicado na imagem. A Figura 3 exemplifica a execução de uma operação de convolução em uma imagem digital $f(x, y)$ com um *kernel* $h(x, y)$.

Figura 3: Exemplo de aplicação de um *kernel* $h(x, y)$ em uma imagem $f(x, y)$ gerando uma nova imagem $g(x, y)$.



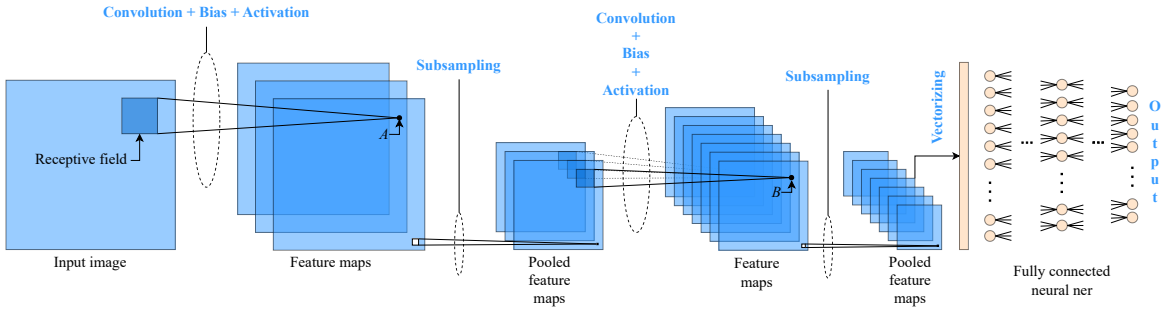
Fonte: Adaptada de Szeliski (2022).

Ao invés de conectarem todos os neurônios de uma camada a outra, uma CNN organiza cada camada em mapas de características (*feature maps*), popularmente chamados de canais. A partir disso, as convoluções em uma rede convolucional de forma geral combinam linearmente as ativações de cada canal de entrada da camada anterior e aplicam diferentes *kernels* para gerar os canais de saída. Além da convolução, os modelos aplicam também, após as camadas convolucionais, operações de *pooling*, as quais são responsáveis por reduzir as dimensões das representações ao longo do processo de treinamento.

As CNNs podem apresentar diferentes arquiteturas, de acordo com as tarefas que pretendem lidar e para aumentar a eficiência do processo de treinamento. A arquitetura da LeNet, por exemplo, mostrada na Figura 4, alterna camadas convolucionais, responsáveis pela extração de características, com camadas de *pooling*, seguidas de camadas totalmente conectadas (*fully connected leyers*), que recebem a linarização das saídas das camadas convolucionais e exibem as respostas do modelo. Especificamente para a tarefa de classificação, a saída da rede ao final das camadas totalmente conectadas consiste

em valores que representam a probabilidade da imagem pertencer a cada uma das classes previstas. A classe que possuir maior probabilidade será a escolhida pelo modelo.

Figura 4: Arquitetura LeNet.



Fonte: Adaptada de Gonzalez (2009).

O treinamento de uma rede convolucional é um processo iterativo baseado em retropropagação (*backpropagation*) e é responsável por ajustar os pesos do modelo. Esse processo envolve duas etapas principais, sendo a primeira a passagem para frente (*forward pass*), onde as saídas de cada camada são calculadas sequencialmente até a camada de saída, e a passagem para trás (*backward pass*), na qual os erros das saídas preditas são calculados e utilizados para corrigir os pesos da rede, propagando essa correção da camada final para as camadas anteriores. Para que o modelo aprenda os melhores pesos para a classificação das imagens, é necessário que esse procedimento seja realizado múltiplas vezes. Cada repetição da retropropagação é denominado um passo do treinamento e cada passagem completa do conjunto de imagens pela rede é chamada de época.

Além disso, é necessário estabelecer uma função de perda (*loss function*), responsável por medir o erro entre as predições e os valores reais, bem como definir um algoritmo de otimização, responsável por realizar os ajustes dos pesos do modelo. Um otimizador comum da literatura é o SGD (*Stochastic Gradient Descendent*), o qual determina a direção do ajuste com base nos gradientes calculados pela retropropagação e a intensidade do ajuste, a partir da taxa de aprendizado estabelecida para o modelo (BOTTOU, 2012).

2.2.1 *Visual Geometry Group Network*

A arquitetura *Visual Geometry Group Network* (VGGNet), proposta por Simonyan e Zisserman (2014), é uma CNN profunda desenvolvida pelo *Visual Geometry Group* da Universidade de Oxford. Ela se destaca por sua estrutura simples, composta por múltiplas camadas convolucionais empilhadas, seguidas por camadas totalmente conectadas. A arquitetura básica dessa rede recebe imagens RGB de tamanho fixo 224×224 *pixels*, com o único pré-processamento sendo a subtração do valor médio de RGB do conjunto de treinamento.

Os autores apresentam como característica diferencial do modelo o uso de filtros de convolução significativamente pequenos, de tamanho 3×3 *pixels*, em todas as camadas convolucionais e com o preenchimento espacial (*padding*) de 1 *pixel* para preservar a resolução. A parte convolucional da rede é dividida em cinco blocos, nos quais, após as convoluções de cada um, aplica-se uma operação de *pooling* espacial com janelas de 2×2. Além disso, todas as camadas ocultas são equipadas com a função de ativação ReLU (*Rectified Linear Unit*) proposta por Krizhevsky, Sutskever e Hinton (2017). Essa função é representada por:

$$f(x) = \max(0, x), \tag{1}$$

a qual apresenta como principal vantagem a velocidade de treinamento em relação às funções de ativação tradicionais como *Sigmoid*. Outra vantagem é que ela não exige normalização de entrada para evitar a saturação no cálculo do gradiente.

A rede também incorporou normalização para auxiliar na generalização e camadas de *pooling*, que reduziram as taxas de erro e ajudaram a prevenir o sobreajuste (*overfitting*). Esse problema é característico de modelos de aprendizado profundo no qual a rede se adapta exageradamente aos dados de treinamento e perde a capacidade de generalização, prejudicando a classificação de novos dados.

Após a pilha de camadas convolucionais, a arquitetura inclui três camadas totalmente conectadas (*Fully Connected* – FC). As duas primeiras camadas FC possuem 4096 neurônios cada, enquanto a última realiza a classificação de acordo com a quantidade de classes do problema em foco. Por fim, última camada do modelo, intitulada

sof-tmax, é responsável por converter a saída da rede em uma distribuição probabilística de pertencimento a cada classe.

Para avaliar o desempenho da arquitetura proposta, os autores testam diferentes configurações da rede, as quais são mostradas na Figura 5, variando principalmente sua profundidade, ou seja, a quantidade de camadas convolucionais. A rede A é a mais rasa, com 11 camadas de peso (8 convolucionais e 3 FC), enquanto a rede E é a mais profunda, com 19 camadas de peso (16 convolucionais e 3 FC). A largura das camadas (número de canais) começa em 64 na primeira camada e aumenta em um fator de 2 após cada camada de *max-pooling*, até atingir 512 canais.

Figura 5: Arquitetura VGGNet.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 x 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Fonte: Adaptada de Simonyan e Zisserman (2014).

Através dos testes, os autores concluíram que a profundidade de uma rede é um fator crítico para o alcance de altas precisões em reconhecimento de imagens em larga escala. Eles demonstraram que, ao aumentar a profundidade para 16–19 camadas, utilizando uma arquitetura com filtros de convolução de tamanho 3×3, é possível obter

uma melhoria significativa em relação às configurações anteriores. Além disso, os autores enfatizaram que as representações visuais aprendidas por suas redes, pré-treinadas no conjunto de dados ImageNet, generalizam-se excepcionalmente bem para uma variedade de outros conjuntos de dados e tarefas de reconhecimento de imagem, onde consistentemente superaram representações menos profundas e abordagens mais complexas.

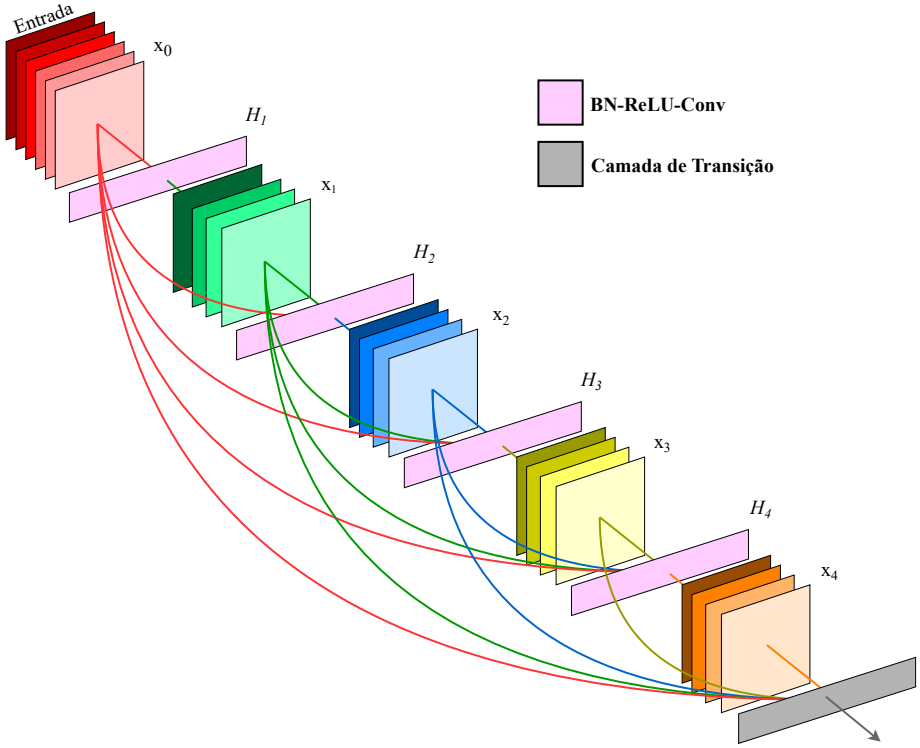
2.2.2 *Densely Connected Convolutional Networks*

A arquitetura *Densely Connected Convolutional Networks* (DenseNet) (HUANG et al., 2017) foi desenvolvida para otimizar o fluxo de informações e gradientes dentro de redes neurais convolucionais profundas. Diferentemente das redes convolucionais tradicionais com l camadas e l conexões, uma entre cada camada com a subsequente, a DenseNet possui $l \times (l+1)/2$ conexões diretas. Especificamente, cada camada l_i recebe como entrada os mapas de características de todas as camadas precedentes $l_0, ..., l_{i-1}$ e, consequentemente, seus próprios mapas de características l_i são utilizados como entradas para todas as camadas subsequentes. Essas combinações de características, mostradas na Figura 6 ocorrem por concatenação, e não por soma como nas ResNets (HE et al., 2016), o que fortalece a propagação das características e incentiva a sua reutilização.

A função de transformação não linear para cada camada, é definida como uma composição de três operações consecutivas, *batch normalization* (BN), seguida por uma unidade linear retificada (*Rectified Linear Unit* - ReLU) e uma convolução de tamanho 3×3 *pixels*. Para lidar com a variação do tamanho dos mapas de características, a rede é estruturada em múltiplos blocos densamente conectados (*dense blocks*), com camadas de transição entre eles. Essas camadas de transição realizam *down-sampling* por meio de uma camada de *batch normalization*, uma convolução de *kernel* 1×1 e um *average pooling* de *kernel* 2×2.

A largura da rede é controlada pelo fator de crescimento (*growth rate*), que determina quantos novos mapas de características são adicionados a cada camada. Essa abordagem permite que a DenseNet alcance um desempenho competitivo em tarefas de classificação de imagens, utilizando menos parâmetros e menos computação do que redes como VGG e ResNet. Neste sentido, sua estrutura facilita o treinamento de redes muito

Figura 6: Exemplo das conexões densas da DenseNet.



Fonte: Adaptada de Huang et al. (2017).

profundas sem a necessidade de mecanismos complexos de regularização. Além disso, os autores exploram variações da arquitetura para otimizar sua precisão e eficiência. Essas variações, mostradas na Figura 7, consistem na alteração da profundidade do modelo, definida pela quantidade de camadas, e de sua largura, variando o parâmetro da taxa de crescimento.

Em suas conclusões, os autores destacam que a DenseNet oferece significativas vantagens. Primeiramente, ela alivia o problema do desaparecimento do gradiente e fortalece a propagação de características através do reaproveitamento das mesmas, o que consequentemente reduz substancialmente o número de parâmetros. Além disso, as conexões densas demonstram um efeito regularizador, amenizando o problema de *overfitting*.

Figura 7: Arquitetura DenseNet.

Layers	Output Size	DenseNet-121 (k = 32)	DenseNet-169 (k = 32)	DenseNet-201 (k = 32)	DenseNet-161 (k = 48)
Convolution	112 x 112	7 x 7 conv, stride 2			
Pooling	56 x 56	3 x 3 max pool, stride 2			
Dense Block (1)	56 x 56	[1 x 1 conv, 3 x 3 conv] x 6	[1 x 1 conv, 3 x 3 conv] x 6	[1 x 1 conv, 3 x 3 conv] x 6	[1 x 1 conv, 3 x 3 conv] x 6
Transition Layer (1)	56 x 56	1 x 1 conv			
	28 x 28	2 x 2 average pool, stride 2			
Dense Block (2)	28 x 28	[1 x 1 conv, 3 x 3 conv] x 12	[1 x 1 conv, 3 x 3 conv] x 12	[1 x 1 conv, 3 x 3 conv] x 12	[1 x 1 conv, 3 x 3 conv] x 12
Transition Layer (2)	28 x 28	1 x 1 conv			
	14 x 14	2 x 2 average pool, stride 2			
Dense Block (3)	14 x 14	[1 x 1 conv, 3 x 3 conv] x 24	[1 x 1 conv, 3 x 3 conv] x 32	[1 x 1 conv, 3 x 3 conv] x 48	[1 x 1 conv, 3 x 3 conv] x 36
Transition Layer (3)	14 x 14	1 x 1 conv			
	7 x 7	2 x 2 average pool, stride 2			
Dense Block (4)	7 x 7	[1 x 1 conv, 3 x 3 conv] x 16	[1 x 1 conv, 3 x 3 conv] x 32	[1 x 1 conv, 3 x 3 conv] x 32	[1 x 1 conv, 3 x 3 conv] x 24
Classification Layer	1 x 1	7 x 7 global average pool			
		1000D fully-connected, softmax			

Fonte: Adaptada de Huang et al. (2017).

2.3 Meta-heurísticas para Solução de Problemas de Otimização

As meta-heurísticas são algoritmos de otimização versáteis e adaptáveis que resolvem de maneira eficaz diversos problemas de otimização. Elas se distinguem por não dependerem de cálculos de derivadas no espaço de busca, ao contrário das técnicas baseadas em gradiente, tornando-as mais flexíveis e capazes de evitar ótimos locais. Além disso, a natureza estocástica desses algoritmos permite que iniciem o processo de otimização gerando resultados aleatórios, o que os ajuda a evitar a convergência prematura e a explorar o espaço de busca de forma eficaz. Em suma, as meta-heurísticas são métodos sofisticados que buscam ou selecionam heurísticas capazes de oferecer a melhor solução possível para um problema de otimização, mesmo com dados incompletos ou recursos computacionais limitados (TOMAR; BANSAL; SINGH, 2024).

Um aspecto fundamental do funcionamento das meta-heurísticas é o equilíbrio dinâmico entre exploração e explotação. A exploração envolve a busca sistemática por novas regiões não mapeadas no espaço de busca para diversificar o conjunto de soluções e aumentar as chances de encontrar soluções novas e potencialmente superiores, o que ajuda a escapar de ótimos locais. Já a explotação, por sua vez, concentra-se em refinar as regiões promissoras do espaço de busca onde posições de alta qualidade já foram identificadas, aprimorando as soluções existentes (BENAISSA et al., 2024). Manter um equilíbrio adequado é crucial para evitar a convergência prematura para soluções subótimas. Essas características tornam esses algoritmos significativamente eficazes para problemas complexos, como os pertencentes à classe de tempo polinomial não-determinístico (*Non Deterministic Polynomial* – NP), os quais são capazes de ser resolvidos em tempo polinomial por algoritmos não-determinísticos.

De forma geral, as meta-heurísticas podem ser divididas em cinco categorias principais, baseadas em suas fontes de inspiração e funcionamento: *Evolution-based*, *Swarm Intelligence-based*, *Physics-based*, *Human-related* e *Hybrid metaheuristic* (TOMAR; BANSAL; SINGH, 2024). Algoritmos baseados na Evolução (*Evolution-based algorithms*) são inspirados na teoria da evolução de Darwin, buscando soluções quase-ótimas através de iterações que incluem seleção parental, recombinação (*crossover*), mutação e seleção de sobreviventes. Não obstante, algoritmos baseados na Inteligência de Enxame (*Swarm Intelligence-based algorithms*) modelam como animais sociais em um rebanho compartilham conhecimento durante o processo de otimização. Algoritmos baseados na Física (*Physics-based algorithms*) replicam princípios físicos da natureza para descobrir a melhor solução. Já algoritmos relacionados a Humanos (*Human-related algorithms*) são impulsionados pela interação social ou padrões de comportamento em pessoas. E os híbridos (*Hybrid metaheuristic*) funcionam a partir da combinação dos operadores mais eficazes de diferentes algoritmos a fim de extrair as melhores características de cada abordagem.

Dentre os principais algoritmos propostos na literatura, é possível citar o algoritmo genético (*Genetic Algorithm* – GA), proposto por Holland (1975). Nele, a população é iniciada de forma aleatória e evolui de acordo com a aptidão dos indivíduos: aqueles com maior aptidão (melhores soluções) são replicados para as próximas gerações, podendo

sofrer mutações ao longo das iterações. Outro exemplo muito utilizado é o algoritmo de otimização por colônia de formigas (*Ant Colony Optimization* – ACO), proposto por Dorigo (1992), no qual o comportamento dos indivíduos se baseia na comunicação através de feromônios comuns em espécies de formigas. As formigas deixam rastros de feromônios ao se deslocarem pelo espaço de busca e tendem a seguir os rastros com níveis mais altos de feromônios (caminho para as melhores soluções).

2.3.1 *Bat-inspired Algorithm*

O método proposto no presente trabalho é baseado em outra meta-heurística, o algoritmo inspirado em morcegos (*Bat-inspired algorithm* – BA), proposto por Yang (2010). Ele se inspira no comportamento dos morcegos de utilizar ecolocalização para se locomover pelo espaço em busca de uma presa. De forma geral, os indivíduos emitem pulsos sonoros altos e ouvem o eco que retorna dos objetos ao seu redor para detectar presas, evitar obstáculos e localizar seus abrigos no escuro. O BA se destaca por sua capacidade de busca diversificada no espaço de soluções, possuindo um comportamento de grupo sofisticado e excelentes capacidades de busca local.

O algoritmo começa com a inicialização aleatória da população de indivíduos. Os morcegos voam aleatoriamente com velocidades v_i e posições x_i, usando uma frequência f_i e volume A_i^0 para encontrar suas presas. A cada iteração, a frequência é escolhida aleatoriamente dentro do intervalo $[f_{min}, f_{max}]$ de acordo com:

$$f_i = f_{min} + (f_{max} - f_{min}) \times \beta, \tag{2}$$

onde β é um valor aleatório pertencente ao intervalo $[0, 1]$. Dependendo da proximidade da presa, cada indivíduo i pode ajustar a emissão de pulso $r_i \in [0, 1]$, aumentando esse valor de acordo com a proximidade da presa. Além disso, o volume A desse pulso varia de valores mais altos para valores menores à medida que se aproxima do objetivo.

A geração de novas soluções é feita com base na atualização das velocidades v_i e posições x_i dos morcegos. A cada iteração t, cada indivíduo atualiza sua velocidade e

posição de acordo com:

$$v_i^t = v_i^{t-1} + (x_i^{t-1} - x_{best}) \times f_i, \tag{3}$$

$$x_i^t = x_i^{t-1} + v_i^t, \tag{4}$$

nas quais x_{best} é a melhor solução atual do algoritmo. Aliado a isso, é realizada uma busca local, selecionando uma solução entre as melhores atuais e gerando uma nova solução com base no volume A_i do morcego e na variância máxima permitida $max(var)$. Essa operação é realizada com base em:

$$x_{new} = x_i^t + (\varepsilon \times A_i^t \times max(var)), \tag{5}$$

onde ε é um valor aleatório no intervalo $[-1, 1]$. À medida que o indivíduo se aproxima de seu objetivo, o volume tende a diminuir e o pulso r_i tende a aumentar, respeitando:

$$A_i^{t+1} = \alpha \times A_i^t, \tag{6}$$

$$r_i^{t+1} = r_i^0 \times [1 - e^{-\gamma \times t}], \tag{7}$$

onde α funciona como um fator de resfriamento e γ é uma constante.

3 Trabalhos Relacionados

Este capítulo fornece uma revisão sobre os principais trabalhos e conceitos envolvendo a otimização de redes neurais convolucionais, como hiperparâmetros e a aestrutura interna. Além disso, apresenta exemplos de estudos que utilizam o algoritmo baseado em morcegos para solucionar esses problemas.

3.1 Otimização de Redes Neurais Convolucionais

A escolha e configuração do modelo de rede convolucional adequado é uma tarefa crítica para tarefas de Visão Computacional. Isso está relacionado ao fato de que cada domínio de aplicação possui suas particularidades e níveis de complexidade. Além de afetar o processo de treinamento e os resultados finais, a aplicação indiscriminada de modelos pré-estabelecidos pode implicar em custo computacional desnecessário. Neste sentido, a busca por modelos cada vez mais eficientes e precisos se tornou um importante campo de pesquisa na área de Visão Computacional. Apesar de novas arquiteturas serem propostas ao longo do tempo, o desenvolvimento de métodos de otimização dos modelos já estabelecidos é um segmento de pesquisa em constante desenvolvimento.

A otimização de hiperparâmetros (*Hyper Parameters Optimization* – HPO) é um aspecto essencial no desenvolvimento de modelos de aprendizado de máquina (*machine learning*), especialmente para CNNs, pois impacta profundamente a robustez, estabilidade e capacidade de generalização dos modelos (WANG; NABNEY; GOLBABAEE, 2024). Ao revelar a influência dos hiperparâmetros no desempenho, a otimização contribui para aumentar a transparência dos modelos de redes neurais, auxiliando os desenvolvedores a tomar decisões mais informadas durante o processo de desenvolvimento do modelo.

O trabalho de Wojciuk et al. (2024) estuda o impacto da HPO na acurácia de classificação de modelos de CNNs ajustados (*fine-tuned*) através de aprendizado por transferência. O estudo busca identificar quais hiperparâmetros são mais importantes, quais seus intervalos ótimos e a viabilidade de usar subconjuntos de dados para a otimização.

Para isso, utilizou o modelo de CNN Xception (CHOLLET, 2017) pré-treinado no ImageNet e três conjuntos de dados públicos, sendo eles CIFAR-100 (KRIZHEVSKY; HINTON et al., 2009), *Stanford Dogs* (KHOSLA et al., 2011) e MIO-TCD (LUO et al., 2018b). Os autores, ao compararem quatro métodos de otimização distintos, mostraram que a HPO pode melhorar a acurácia da classificação da CNN em até 6%.

Além dos hiperparâmetros, a estrutura interna de uma rede convolucional também afeta diretamente o desempenho e eficiência do modelo. Em Gonçalves, Souza e Fernandes (2022), os autores buscam otimizar a arquitetura das camadas totalmente conectadas de redes convolucionais para a detecção precoce de câncer de mama usando imagens infravermelhas estáticas. O estudo aplica os algoritmos bioinspirados *Genetic Algorithm* (GA) e *Particle Swarm Optimization* (PSO) a três CNNs, VGG-16, ResNet-50 e DenseNet-201, usando a técnica de transferência de aprendizado, onde apenas as camadas FC são otimizadas, enquanto as camadas convolucionais pré-treinadas são mantidas. A base de dados utilizada foi DMR-IR (SILVA et al., 2014) e os resultados mostraram que o método proposto foi capaz de melhorar os modelos em comparação com as configurações manuais. Por exemplo, a acurácia da VGG-16 foi aprimorada em quase 30% e a ResNet-50 teve um aumento de acurácia de quase 20%.

3.1.1 Otimização da Estrutura

As redes convolucionais são caracterizadas por três dimensões fundamentais: profundidade, largura e resolução, as quais são determinantes para sua capacidade de processar os dados de entrada. A profundidade corresponde à quantidade de camadas presentes na arquitetura. Modelos mais profundos são capazes de extrair características mais sofisticadas e complexas, além de aprender transformações mais elaboradas, proporcionando maior capacidade preditiva em relação aos modelos compostos por uma única camada. Contudo, o acréscimo excessivo de camadas pode resultar em ganhos limitados na acurácia do modelo.

Por sua vez, a largura está associada ao número de neurônios ou canais existentes em cada camada. Arquiteturas mais largas tendem a capturar informações mais detalhadas e refinadas, possibilitando o aprendizado de padrões diversos em diferentes direções e

escalas, além de, geralmente, serem mais simples de treinar. Entretanto, redes exageradamente largas e pouco profundas podem apresentar limitações na identificação de padrões de alto nível (TAN; LE, 2019). A resolução, por fim, refere-se ao tamanho da imagem de entrada. Imagens de maior resolução permitem que as redes convolucionais extraiam detalhes mais precisos, o que pode contribuir para o aumento da acurácia. No entanto, esses ganhos tendem a ser menos expressivos quando se trabalha com resoluções extremamente elevadas. Tradicionalmente, a construção de arquiteturas de redes de aprendizado profundo era baseada em disponibilidade fixa de recursos computacionais, as quais eram escaladas de maneira arbitrária para obter melhor acurácia dentro dos limites de custo computacional. Esse processo, de forma geral, envolvia a modificação isolada de uma das três dimensões a fim de avaliar o impacto das modificações no desempenho do modelo.

Um exemplo é o trabalho proposto por Li (2024), o qual investiga o efeito da profundidade e largura de modelos de CNN na classificação binária de imagens de cães e gatos. O método proposto envolve a criação de dois tipos principais de arquiteturas, uma profunda, com múltiplas camadas ocultas, e uma rasa e larga com uma única camada oculta, mas com um número maior de neurônios. Através dos experimentos, os autores avaliaram que redes excessivamente profundas levam a um declínio na precisão devido ao problema de perda de gradiente, indicando que uma profundidade ideal deve ser buscada para um desempenho ótimo. Além disso, ao variar o número de neurônios em uma única camada oculta, a precisão da segunda rede aumentou até um certo ponto. Porém acima de um limiar, a precisão diminuiu significativamente, acompanhada por um aumento no tempo de treinamento.

De forma semelhante, o trabalho Jara, Sheta e Elashmawi (2024) apresenta uma arquitetura para o diagnóstico precoce do câncer de mama utilizando imagens de ultrassom. O método proposto utiliza um conjunto de dados público de imagens de ultrassom de mama, que inclui classificações como malignas, benignas e normais, com o objetivo de avaliar o impacto da resolução das imagens de entrada no desempenho do modelo. De acordo com os autores, a acurácia geral do modelo aumentou com a resolução da imagem, atingindo um valor acima 83% para os dados de teste com a resolução de 256×256 *pixels*. A acurácia do modelo nessa resolução foi melhor em comparação com as resoluções de

32×32, 56×56 e 128×128. No entanto, a melhora na acurácia de 56×56 para 128×128 foi mais significativa do que de 128×128 para 256×256, sugerindo que o benefício do aumento da resolução diminui após um certo ponto.

Tendo em vista os dois trabalhos apresentados, destaca-se que em ambos, o processo de escalonamento das dimensões da rede foi arbitrário e manual, o que se configura como uma tarefa cansativa e repetitiva, além de ser suscetível a enviesamentos dos testes por parte de concepções prévias dos pesquisadores. Para solucionar isso, Tan e Le (2019) propõem o método de escalonamento composto (*compound scaling method*), o qual se baseia na necessidade de se equilibrar cuidadosamente as três dimensões da rede para obter um desempenho superior. Nele, as três dimensões são aumentadas uniformemente através de um coeficiente composto ϕ, o qual controla a quantidade de recursos adicionais disponíveis para serem distribuídos entre a profundidade, largura e resolução da arquitetura.

Os autores estudaram sistematicamente a escalabilidade de uma CNN e observaram que cada uma das dimensões possui um impacto diferente no cálculo da quantidade de operações de ponto flutuantes (*Floating Point Operations* – FLOPs) realizadas pelo modelo. A quantidade de FLOPs é diretamente proporcional à profundidade, o que significa que se o número de camadas duplicar, as operações também duplicam. Já em relação à largura e resolução, a quantidade de operações é proporcional ao quadrado dessas dimensões, ou seja, se uma delas duplicar, a quantidade de FLOPs quadruplica. Neste sentido, os autores definem que o escalonamento do modelo através do coeficiente ϕ aumenta a quantidade de FLOPs de acordo com:

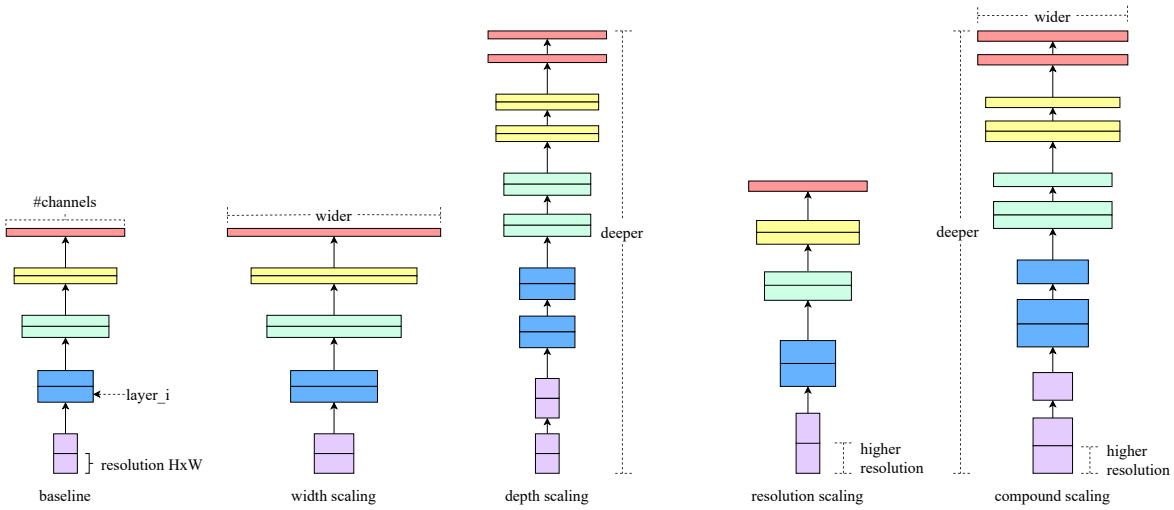
$$FLOPs^{\phi} = (\alpha \times \beta^2 \times \gamma^2)^{\phi}, \tag{8}$$

onde α representa o coeficiente de escalonamento da profundidade, β o da largura e γ o da resolução da imagem.

Para avaliar a eficácia do método, os autores projetaram uma nova arquitetura chamada EfficientNet-B0 e aplicaram o escalonamento composto para desenvolver as variações B1 até B7. Inicialmente, fixaram o valor de ϕ em 1 com o objetivo de encontrar os melhores valores para α, β e γ, através de uma busca em grade (*grid-search*). Posteri-

ormente, a rede B0 é escalonada com diferentes valores de ϕ onde as novas dimensões de profundidade, largura e resolução são dadas respectivamente por α^{ϕ}, β^{ϕ} e γ^{ϕ}. Os autores destacam que essa metodologia de duas etapas é eficiente, pois a busca mais cara pelos coeficientes α, β e γ é realizada apenas uma vez na rede base. Além disso, o escalonamento composto permite um balanceamento uniforme e otimizado da profundidade, largura e resolução da rede, ao invés de escalar apenas uma dimensão de forma arbitrária, o que tem se mostrado mais eficaz para obter maior precisão e eficiência. A Figura 8 mostra o resultado visual do *compound scaling*.

Figura 8: Representação visual do método *Compound Scaling*.



Fonte: Adaptada de Tan e Le (2019).

3.1.2 Algoritmo Inspirado em Morcegos para otimização de CNNs

O algoritmo BA é amplamente empregado em diferentes contextos, inclusive na área de Visão Computacional para otimização de aspectos de redes de aprendizado profundo. Ravikiran et al. (2024), por exemplo, propõem uma abordagem que utiliza o BA para otimizar hiperparâmetros críticos de uma CNN para classificação de raças de ovelhas. Eles focam especificamente em otimizar a taxa de aprendizado e a taxa de *dropout* e os resultados alcançaram uma acurácia de teste de 96%, superando outros métodos da literatura como VGG16 com 95,80% e MobileNetV2 com 95%. Outro exemplo é o trabalho de Bacanin et al. (2020), no qual propõe-se uma otimização baseada em morcegos especificamente para a taxa de *dropout*. Nele, os autores testam a precisão da rede em conjuntos

de dados comuns da literatura como MNIST e CIFAR-10, comparando o desempenho do BA com outras meta-heurísticas como *Particle Swarm Optimization* (PSO), *Firefly Algorithm* (FA) e *Cuckoo Search* (CS). Os resultados mostraram que o método proposto superou os outros algoritmos em ambos os conjuntos de dados, com acurácia de 71,76% no CIFAR-10 e 99,19% no MNIST.

Tendo em vista o desempenho e variedade de possibilidades de aplicações do BA, o presente trabalho possui como proposta principal utilizá-lo para a otimização das dimensões de uma rede neural convolucional, especificamente para a profundidade e a largura. Esse objetivo foi definido tendo em vista o que foi discutido anteriormente neste capítulo, no qual destacou-se que o dimensionamento adequado do modelo é essencial para a obtenção de precisão superior com um consumo de recursos equilibrado. Os detalhes do método construído são descritos no Capítulo 5.

4 Conjunto de Dados

A construção da base de dados é uma etapa crítica para o desenvolvimento de soluções de Visão Computacional. Luo et al. (2018a) mostram que existe uma correlação positiva entre a quantidade de instâncias e o desempenho do modelo, de forma que quanto mais bem representadas são as classes do problema, ou seja, quanto maior a quantidade de instâncias na base, melhores serão os resultados. Além da quantidade, a escolha de imagens que representem de forma fidedigna cada objeto é essencial, uma vez que, como discutido anteriormente, as classes em um problema de classificação fina possuem diferenças sutis e quase imperceptíveis ao olho humano. Tendo isso em vista, este capítulo descreve os detalhes sobre o conjunto de dados utilizado neste trabalho.

4.1 Descrição dos Dados

O presente trabalho faz uso de uma base de dados construída no contexto do projeto *Happy Cow ID*, o qual pertence ao projeto "Residência Zootécnica Digital" desenvolvido pela Embrapa Gado de Leite em parceria com instituições de ensino superior, como a Universidade Federal de Juiz de Fora. Todas as imagens foram coletadas manualmente em fazendas por alunos pertencentes ao projeto, sob orientação de um profissional da Embrapa Gado de Leite. Isso foi feito com o objetivo de obter a representação dos animais em cenários do mundo real, trazendo uma melhor representação dos indivíduos e uma classificação mais precisa.

Nomeado como *"Complete Dataset"*, o conjunto de dados consiste em 4071 imagens de bovinos divididos entre as raças "Girolando" e "Holandês", como visto na Figura 9, de acordo com as proporções mostradas na Tabela 1. As imagens são coloridas, pertencendo ao espectro RGB e possuem diferentes resoluções. Além disso, a base conta com um total de 88 indivíduos diferentes representados nas posições frontal, lateral e diagonal, ou seja, existem múltiplas imagens de cada animal capturado. Isso foi feito com o objetivo de possibilitar a obtenção de informações relevantes para a classificação que podem ficar

ocultas em certos ângulos de visualização.

Figura 9: Exemplos de imagens de cada raça abordada.

(a) Girolando

(b) Holandês

Fonte: Elaborada pelo autor (2025).

Tabela 1: Quantidades de imagens nos conjunto de dados construídos.

Conjunto	Girolando	Holandês	Total
Complete Dataset	1954	2117	4071
Optimization Set	588	621	1209

Fonte: Elaborada pelo autor (2025).

Tendo em vista o foco do projeto *Happy Cow ID* em analisar as expressões faciais dos animais, todas as capturas passaram por um pré-processamento, nas quais foi realizado o recorte facial dos bovinos. Essa operação foi feita com a ajuda da ferramenta de anotação LabelImg[1] na versão 1.8.6, utilizada para a anotação de objetos para o problema de detecção. Nela, desenhou-se as caixas delimitadoras que enquadram a face do animal, e a partir das coordenadas salvas nos arquivos de saída da anotação, realizou-se a operação de recorte nas imagens originais. Após o recorte, as imagens apresentaram resoluções

[1]https://github.com/HumanSignal/labelImg

variadas de 454×454 *pixels* a 4000×4000 *pixels*. Um exemplo de uma imagem antes e depois do recorte facial pode ser visto na Figura 10.

Figura 10: Exemplo de recorte facial aplicado a uma imagem de bovino.

(a) Imagem completa

(b) Recorte facial

Fonte: Elaborada pelo autor (2025).

A fim de otimizar o custo de tempo envolvido na execução do algoritmo de otimização baseado em morcegos, foi construído um conjunto, denominado "*Optimization Set*", a partir da amostragem de aproximadamente 30% das imagens contidas no *Complete Dataset*. Esse subconjunto foi estabelecido respeitando as proporções de desbalanceamento existentes no conjunto de dados principal, como mostra a Tabela 1, e visa estabelecer uma aproximação do desempenho dos modelos no problema de classificação de raças de bovinos. Tendo em vista a natureza iterativa do algoritmo BA, considerando que o número de treinamentos realizados é diretamente proporcional ao número de iterações e ao tamanho da população, a construção desse conjunto foi essencial para amenizar o tempo gasto durante os experimentos realizados.

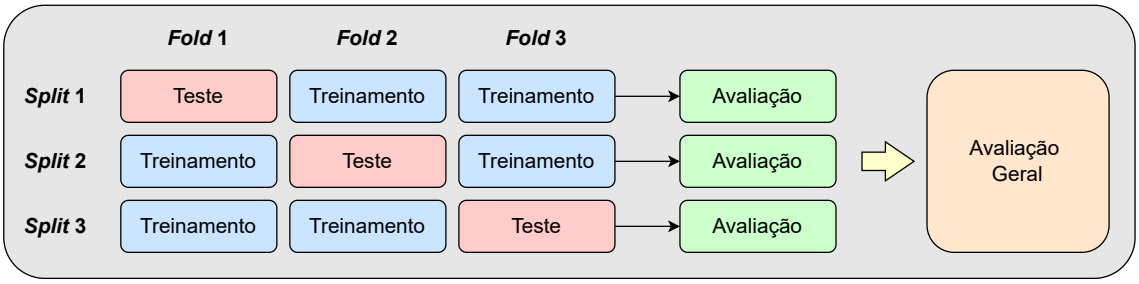
4.2 Organização dos Dados para Treinamento

Na área de aprendizado de máquina supervisionado, os modelos treinados podem sofrer com um problema comum denominado *overfitting*. Nesse problema, o processo de treinamento não é capaz de generalizar para dados não observados anteriormente, ou seja, há um sobreajuste nos dados observados durante o treino e uma deficiência durante a classificação de dados de teste. De acordo com Ying (2019), uma das principais causas desse problema é a aprendizagem de ruído no conjunto de treinamento. Quando esse con-

junto é de tamanho insuficiente, possui dados pouco representativos ou apresenta muito ruído, o modelo, por consequência, é afetado por essas inconsistências, prejudicando seu desempenho em cenários do mundo real. Não obstante disso, uma das formas de amenizar a ocorrência de *overfitting*, é a manutenção da qualidade da base, através do aumento na quantidade de imagens, buscando dados mais representativos entre as classes e aplicando estratégias de treinamento aprimoradas.

Tendo em vista a natureza da classificação de raças de bovinos, a qual se configura como uma classificação fina, como discutido anteriormente, entende-se que o conjunto de dados se mostra um importante parâmetro de ajuste a fim de evitar a ocorrência de *overfitting* e garantir o desempenho do treinamento. Neste caso, optou-se por modelar o conjunto de dados para os experimentos deste trabalho seguindo o método de validação cruzada. A ideia central desse método é dividir igualmente o conjunto de imagens em k subconjuntos chamados *folds* (QIU, 2024). Para cada *fold* i, onde $1 \leq i \leq k$, constrói-se uma nova configuração de treino-teste (*split* i), onde o *fold* i pertence ao conjunto de teste e o restante dos *folds* pertencem ao conjunto de treinamento. Esse processo garante que cada imagem tenha a chance de estar tanto no conjunto de treinamento quanto no de teste, proporcionando uma avaliação de desempenho mais completa e menos sujeita a viés do que uma única divisão de treino-teste. A Figura 11 mostra um exemplo de uma validação cruzada com $k = 3$ (3-*fold*).

Figura 11: Exemplo de um 3-*fold*.



Fonte: Elaborada pelo autor (2025).

Neste trabalho, optou-se pela implementação de uma validação cruzada do tipo 3-*fold*, na qual os dados são divididos em três partes aproximadamente iguais, contendo cerca de $33,33\%$ do total cada. Em cada iteração (*split*), dois *folds*, aproximadamente $66,67\%$ dos dados, são utilizados para treinamento e o *fold* restante, aproximadamente

33, 33%, é designado para teste. Em adição, reservou-se 20% do conjunto de treino de cada (*split i*) para validação, o que representa aproximadamente 16% to total dos dados.

Como mostrado na Tabela 1, apesar de ambas as classes apresentarem um quantidade significativa de imagens, ainda há um pequeno desbalanceamento entre elas. Neste caso, a separação entre os *folds* foi feita de maneira estratificada, preservando o desbalanceamento das classes entre cada um dos subconjuntos construídos. Além disso, como cada um dos indivíduos capturados possui mais de uma imagem, durante o processo de estruturação da validação cruzada, realizou-se como uma etapa adicional de estratificação a aglomeração de todas as imagens de cada animal no mesmo *fold*. Isso foi feito com objetivo de evitar possíveis enviezamentos de treinamento, uma vez que, se uma vaca possuir, ao mesmo tempo, imagens nos conjuntos de treinamento e teste, o modelo tenderá a classificá-la corretamente, já que esse animal já foi visto no conjunto de treino. A Tabela 2 mostra a configuração de cada um dos *splits* com as respectivas quantidades de imagens de cada raça e sua representação percentual do total.

Tabela 2: Configuração do 3-*fold* construído.

Split	Treinamento			Validação			Teste		
	Girolando	Holandês	%	Girolando	Holandês	%	Girolando	Holandês	%
1	1053	1089	52,62	267	269	13,16	634	759	34,22
2	1094	1113	54,22	215	336	13,53	645	668	32,25
3	997	1086	51,17	282	341	15,30	675	690	33,53

Fonte: Elaborada pelo autor (2025).

5 Abordagem Proposta

Este capítulo foca em apresentar os detalhes e características do método desenvolvido, o qual compreende a modelagem da estrutura das redes neurais utilizadas e a metodologia do algoritmo de otimização baseado em morcegos utilizado para dimensionar os modelos.

5.1 Modelos de Classificação de Imagens

Este trabalho fez uso de duas redes neurais para classificação de imagens. A primeira consiste em uma variação da arquitetura VGGNet e a segunda uma variação da arquitetura DenseNet, ambas detalhadas na Seção 2.2. Elas foram escolhidas com o objetivo de comparar como diferentes propostas de arquitetura interferem no problema abordado. Enquanto a VGGNet se configura como uma rede relativamente simples, composta pelo encadeamento de blocos convolucionais, a DenseNet apresenta uma abordagem mais sofisticada, a partir da elaboração de blocos convolucionais densos que recebem como entrada a concatenação das saídas dos blocos anteriores. Além disso, tendo em vista a proposta do método, que consiste em avaliar como diferentes combinações das dimensões das redes influenciam no aprendizado do modelo, estabeleceu-se uma configuração padrão para ambas as arquiteturas. Essas configurações foram usadas como *baseline* de comparação para as redes personalizadas encontradas pelo algoritmo de otimização.

Para o *baseline* da VGGNet, escolheu-se uma variação composta por 11 camadas convolucionais, representada pela Figura 12a. Essas camadas, que representam a profundidade da rede, são divididas em 5 blocos, nos quais, cada um aplica uma quantidade fixa de operações de convolução de *kernel* 3×3 seguidas de uma operação de *pooling* máximo (*maxpool*) 2×2. Os blocos 1 a 5 possuem respectivamente 3, 2, 2, 2 e 2 camadas de convolução. Já a largura da rede foi escolhida segundo os valores padrões estabelecidos pelos autores, em que cada um dos 5 blocos possuem respectivamente 64, 128, 256, 512 e 512 canais de saída. Após a parte convolucional, são adicionadas 3 camadas totalmente conectadas, em que as duas primeiras possuem 4096 canais de saída e a terceira exibe os

resultados para o problema de classificação binária.

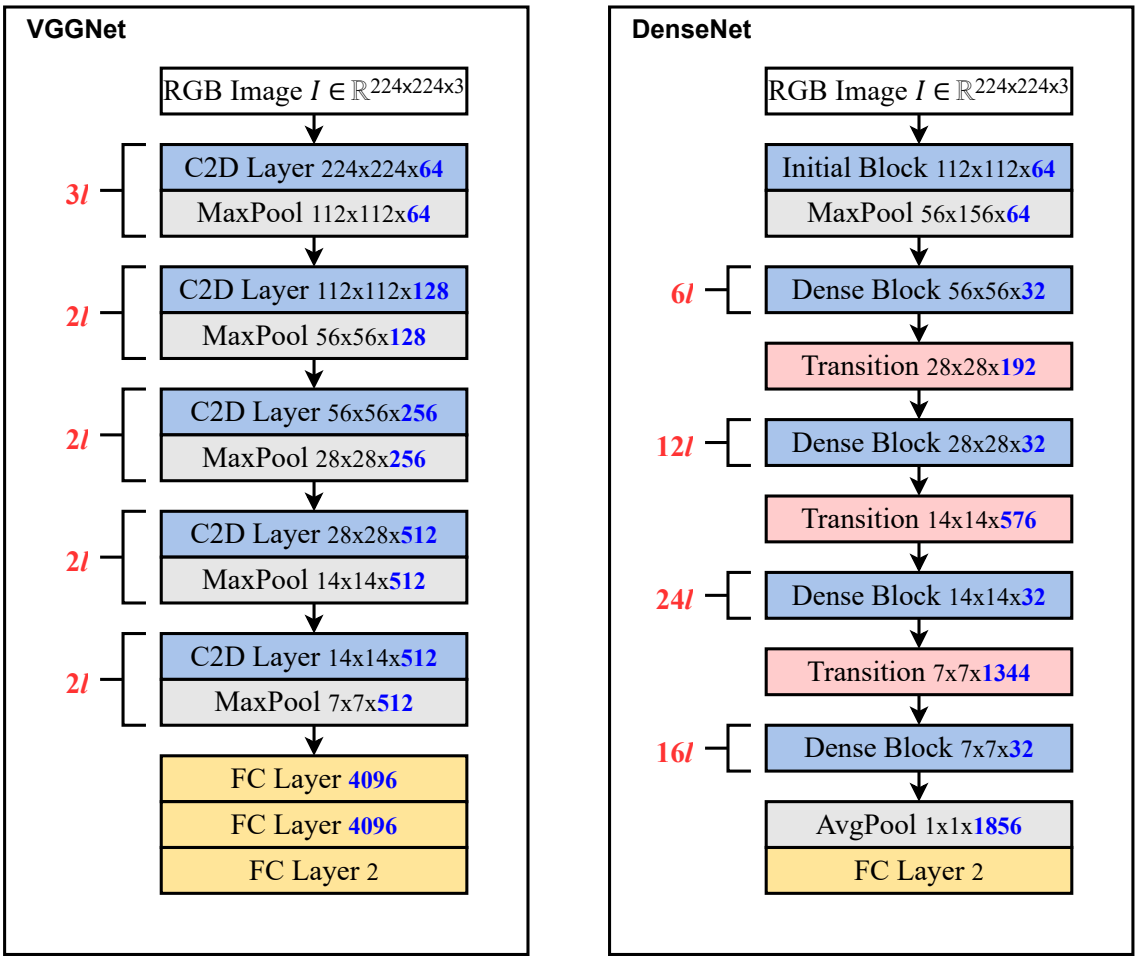
O *baseline* da DenseNet, mostrado na Figura 12b, corresponde à uma variação popular da arquitetura composta por 121 camadas. Essas camadas são divididas por padrão em 4 blocos densos intercalados por blocos de transição. Cada bloco denso compreende um conjunto de camadas densas, as quais são construídas a partir da composição de uma operação de convolução com *kernel* de tamanho 1×1 com uma convolução 3x3. Nesta configuração, cada bloco possui respectivamente 6, 12, 24 e 16 camadas densas, o que totaliza 116 camadas convolucionais. As camadas de transição apresentam uma convolução 1×1 e um *pooling* médio (*AvgPool*) 2×2. Além disso, o modelo se inicia com um bloco inicial composto por convolução de *kernel* 7×7 seguido por um *pooling* máximo 3×3. Após a parte convolucional, é aplicado um *pooling* médio 1×1 seguido por uma camada totalmente conectada que exibe os resultados da classificação. Por fim, a largura da rede é definida pela taxa de crescimento $k = 32$, fazendo com que cada bloco denso exiba como saída 32 canais e as camadas de transição apresentam respectivamente 192, 576 e 1344 canais de saída.

A fim de realizar uma comparação justa entre o desempenho das duas arquiteturas, todos os experimentos foram realizados com a mesma configuração da base de dados. Aliado a isso, tendo em vista que o método altera diretamente a estrutura dos modelos, ambas as redes foram treinadas sem a utilização de pesos pré-treinados em outros conjunto de dados. Por fim, fixou-se a resolução das imagens de entrada em 224×224 *pixels* para os dois modelos. Neste caso, os *feature maps* de saída da VGGNet e DenseNet são respectivamente de resoluções 7×7 e 1×1.

5.2 Algoritmo de Otimização

O objetivo principal deste trabalho é entender como o dimensionamento de uma rede convolucional afeta o desempenho do modelo em um problema de classificação, que, neste contexto, é a classificação de raças de bovinos. Para isso, propõe-se a modelagem e implementação de uma meta-heurística baseada em morcegos capaz de automatizar o escalonamento da profundidade e largura de uma arquitetura, equilibrando o consumo de recursos com a qualidade dos resultados. Tendo isso em vista, esta seção detalha a

Figura 12: Arquiteturas das redes estabelecidas como *baseline* de comparação.



(a) VGGNet *baseline*. (b) DenseNet *baseline*.

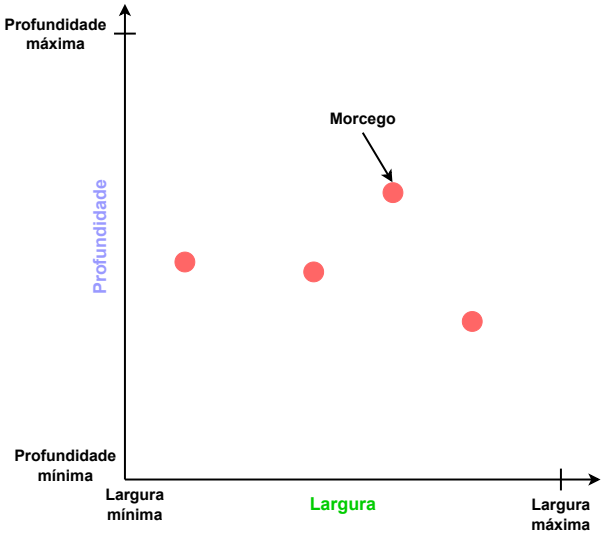
Fonte: Elaborada pelo autor (2025).

modelagem e o funcionamento do método desenvolvido.

Como discutido na Seção 2.3.1, o BA consiste em construir uma população posicionada aleatoriamente no espaço de busca e atualizá-la iterativamente a partir da atualização das frequências e velocidades dos morcegos. Diferentemente da versão original, na presente pesquisa há a necessidade de otimizar dois parâmetros distintos. Neste caso, pode-se definir o espaço de busca do algoritmo como uma representação bidimensional, exemplificada na Figura 13, onde a posição de cada morcego é definida simultaneamente pelos valores profundidade e largura da rede. Essa natureza multi-parâmetro do problema aumenta significativamente a dificuldade de encontrar soluções relevantes, uma vez que, considerando os intervalos definidos para profundidade e largura respectivamente $[x_{min}, x_{max}]$ e $[y_{min}, y_{max}]$, a quantidade de combinações z possíveis entre os parâmetros é

dada por $z = (x_{max} - x_{min}) \times (y_{max} - y_{min})$.

Figura 13: Espaço de busca multi-parâmetros.



Fonte: Elaborada pelo autor (2025).

Para definir os limites do espaço de busca do algoritmo, utilizou-se a ideia proposta por Tan e Le (2019), discutida na Seção 3.1.1. De acordo com os autores, a quantidade de FLOPs aumenta proporcionalmente ao aumento das três dimensões principais da rede, de acordo com a Equação 8. A partir dela, fixando $\phi = 1$, entende-se que, assumindo $FLOPs = 1$ como sendo o fator que representa a quantidade de FLOPs na rede *baseline*, os escalonadores das três dimensões devem assumir necessariamente os valores $\alpha = 1, \beta = 1$ e $\gamma = 1$. Neste caso, fixando $FLOPs = 2$, por exemplo, ou seja, aumentando a tolerância para o dobro de operações de ponto flutuante, os escalonadores α, β e γ podem ter seus valores aumentados ou diminuídos proporcionalmente de forma a atingir essa quantidade de FLOPs.

Tendo isso em vista, o limite máximo do espaço de busca para as duas redes utilizadas é calculado com base na tolerância à quantidade de FLOPs da seguinte forma: inicialmente, a fim de simplificar os cálculos, assumiu-se que o valor máximo s para os três escalonadores seria o mesmo, de acordo com:

$$\alpha_{max} = \beta_{max} = \gamma_{max} = s. \tag{9}$$

Dado que este trabalho foca apenas na otimização da profundidade e da largura,

o valor da resolução é fixo em 224×224 *pixels* para ambas as redes, ou seja, o escalonador de resolução γ também será fixado em 1. Isso faz com que a equação original dos autores de Tan e Le (2019) possa ser modificada para a forma descrita a seguir:

$$FLOPs_{max} = (s \times s^2 \times s^2) = (s \times s^2 \times 1). \tag{10}$$

Manipulando uma última vez a equação, encontra-se o valor máximo dos escalonadores s em função da quantidade máxima de operações ($FLOPS_{max}$):

$$s = (FLOPs_{max})^{\frac{1}{3}}. \tag{11}$$

Por fim, os valores mínimos das dimensões foram escolhidos arbitrariamente para ambas as redes com o objetivo de respeitar a arquitetura proposta. Para a VGGNet, os valores mínimos dos escalonadores de profundidade e largura são, respectivamente, $\alpha = 0,50$ e $\beta = 0,20$. O valor de α foi escolhido com o objetivo de manter a estrutura de 5 blocos, com pelo menos uma camada convolucional em cada um. Já para a DenseNet, como se trata de uma rede significativamente mais profunda, os valores foram definidos, respectivamente, como $\alpha = 0,20$ e $\beta = 0,20$, possibilitando a construção de redes mais rasas.

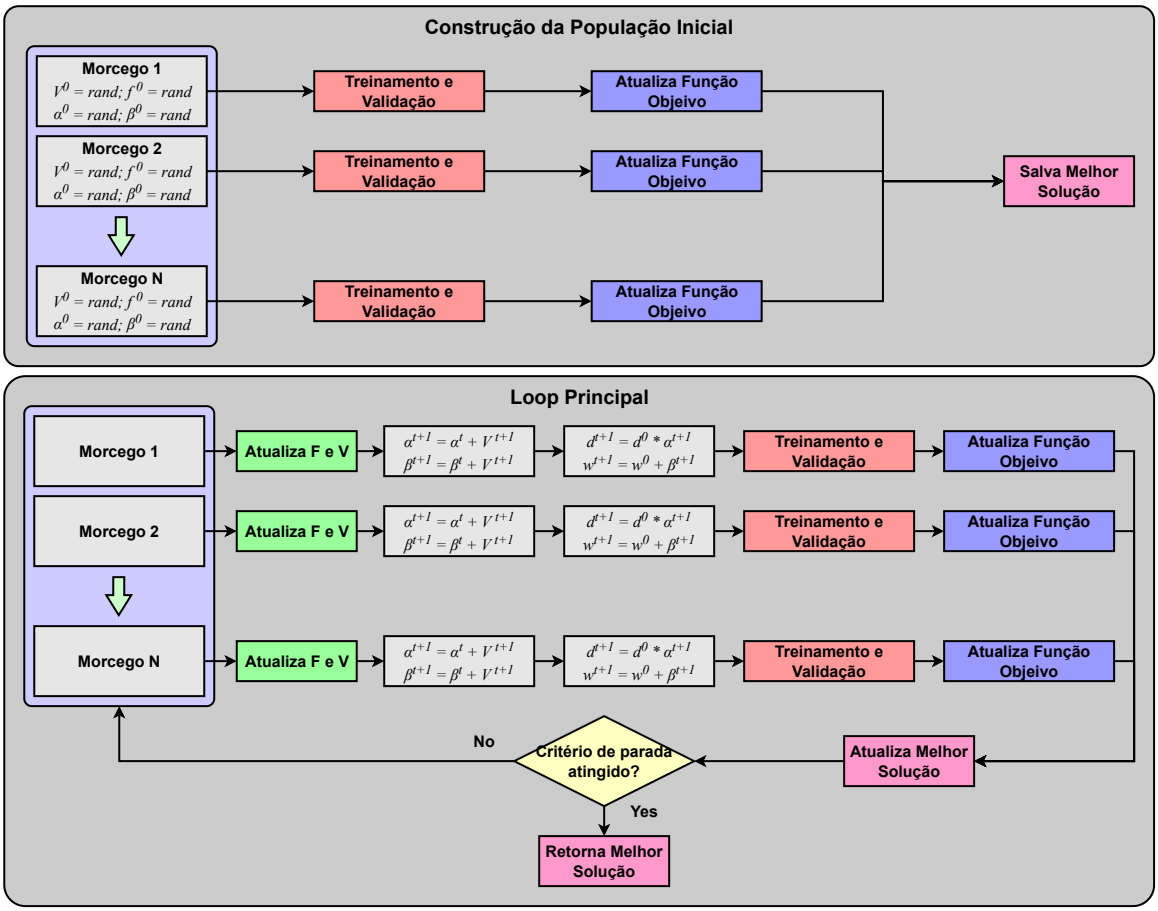
O funcionamento do algoritmo é dividido em duas etapas principais: a construção da população inicial e o *loop* principal, como descrito no diagrama da Figura 14. A população inicial é definida com n morcegos em posições aleatórias do espaço de busca, ou seja, com valores aleatórios dos escalonadores α e β. Cada morcego possui duas velocidades, uma para cada dimensão da rede, as quais são inicializadas com uma fração da posição inicial de cada componente:

$$V_i^0 = random(h_0, h_1) \times x_i^0, \tag{12}$$

onde $0 \leq h_0 < h_1 \leq 1$ são valores aleatórios e x_0 é o valor inicial da respectiva componente. Os indivíduos também possuem frequências separadas para cada dimensão da rede, em que cada uma é inicializada de forma aleatória respeitando os valores máximos e mínimos,

como descrito no algoritmo original. Após a inicialização dos morcegos, as suas respectivas arquiteturas serão construídas e treinadas na base de dados *Optimization Set*, atualizando seus respectivos valores da função objetivo. A melhor solução inicial será escolhida como o melhor global inicial.

Figura 14: *Pipeline* do algoritmo de otimização BA.



Fonte: Elaborada pelo autor (2025).

Após a construção da população inicial, o *loop* principal se inicia, seguindo o raciocínio descrito no algoritmo original. Em cada iteração, os morcegos atualizam suas frequências, velocidades e posições de acordo com as Equações (2), (3) e (4), e utilizam os valores calculados para as posições de cada componente (α e β) para definir as novas dimensões da arquitetura:

$$Depth_i^t = int(Depth_{baseline} \times \alpha_i^t), \tag{13}$$

$$Width_i^t = int(Width_{baseline} \times \beta_i^t), \tag{14}$$

onde as novas dimensões das arquiteturas são definidas a partir da multiplicação das dimensões originais pelos respectivos valores dos escalonadores da iteração atual. Como a profundidade e largura das redes são definidas por valores inteiros, o resultado das multiplicações nas equações 13 e 14 são arredondados para o inteiro mais próximo. Ao atingir o critério de parada, o qual foi definido arbitrariamente como o total de iterações, a melhor arquitetura encontrada pelo algoritmo é selecionada para o treinamento na base de dados *Complete Dataset*.

A função objetivo construída neste trabalho tem o objetivo de equilibrar os resultados da classificação com o consumo de recursos computacionais do modelo. Testes preliminares mostraram que utilizar apenas as informações de FLOPs, apesar de ter impacto em ambas as dimensões, afeta a profundidade mais significativamente do que a largura. Portanto, acrescentou-se as informações da quantidade de parâmetros aprendíveis pelo modelo como componente da função. De forma geral, o objetivo é maximizar a acurácia da rede enquanto minimiza o número de parâmetros e o número de operações. A acurácia do modelo é dada pela acurácia do subconjunto de validação do conjunto de dados "*Optimization Set*".

O formato final da função objetivo é:

$$Objective_i^t = \frac{B_1 \times Acc_i^t}{1 + (B_2 \times FLOPS_{ratio}) + (B_3 \times Params_{ratio})}, \tag{15}$$

onde B_1, B_2 e B_3 são respectivamente os pesos atrelados às componentes Acc_i^t, que correspondem à acurácia do modelo construído pelo morcego i na iteração t, $FLOPS_{ratio}$ e $Params_{ratio}$. O consumo de FLOPs ($FLOPS_{ratio}$) e parâmetros ($Params_{ratio}$) é dado pela razão entre o valor dessas componentes no modelo *baseline* e seu valor corrente no algoritmo de otimização:

$$FLOPs_{ratio} = \frac{FLOPs_i^t}{FLOPs_{baseline}}, \tag{16}$$

$$Params_{ratio} = \frac{Params_i^t}{Params_{baseline}}. \tag{17}$$

Além disso, somou-se 1 ao cálculo do consumo geral no denominador da função com o objetivo de amenizar o impacto dessas razões de consumo no resultado final da

função. Em testes anteriores, observou-se que o consumo de recursos em relação ao *baseline* possuía uma variação significativamente maior que a variação das acurácias, o que fazia com que o algoritmo de otimização desse prioridade exclusiva às componentes do denominador, derrubando as dimensões da rede sem levar em consideração a queda da acurácia. Ao adicionar 1 ao denominador, a função objetivo fica limitada ao intervalo $[0, B_1 \times Acc_i^t]$, ou seja, à medida que os morcegos diminuem as dimensões da rede, as componentes de consumo de FLOPs e parâmetros tendem a perder relevância, fazendo com que o algoritmo priorize a acurácia. Isso evita que o BA foque em redes mínimas e de baixa acurácia, solucionando o problema.

6 Experimentos e Resultados

Para avaliar a eficácia do método proposto, foi conduzido um conjunto de experimentos de otimização e treinamento, utilizando a base de dados construída e as arquiteturas previamente selecionadas. Este capítulo apresenta as configurações adotadas nesses experimentos, bem como os resultados obtidos para as melhores combinações de parâmetros do algoritmo BA e das arquiteturas dos modelos mais promissores. Além disso, são discutidas as principais implicações do método, suas conclusões, pontos fortes, limitações e possíveis direções para trabalhos futuros.

6.1 Configuração dos Experimentos

Os experimentos realizados neste trabalho fizeram uso de uma máquina pertencente à Rede Integrada de Pesquisa em Alta Velocidade (RePesq)[2] da UFJF, cujas configurações são exibidas na Tabela 3. Os experimentos fizeram uso tanto do processador, responsável pela execução do algoritmo de otimização, quanto da placa de vídeo, na qual os modelos foram treinados. Além disso, os códigos desenvolvidos fizeram uso da linguagem *Python*[3] na versão 3.13.2 e da biblioteca *Pytorch*[4] na versão 2.2.2, para implementação e treinamento dos modelos.

Tabela 3: Configurações da máquina utilizada.

Memória RAM	Placa de Vídeo	Armazenamento	Sistema Operacional
8GB	NVIDIA A30 24GB	64GB	Ubuntu 22.04.5 LTS

Fonte: Elaborada pelo autor (2025).

6.1.1 Otimização dos Modelos

O processo de otimização dos *baselines* foi conduzido por meio de experimentos que variaram os principais parâmetros do algoritmo baseado em morcegos. Dentre esses, o pulso

[2]https://www.repesq.ufjf.br/
[3]https://www.python.org/
[4]https://pytorch.org/

e o volume são particularmente relevantes, pois controlam diretamente o equilíbrio entre exploração e refinamento local durante a busca por soluções. Inicialmente, buscou-se identificar a melhor combinação entre esses dois parâmetros, delimitando os intervalos possíveis para a realização da busca local. Essa etapa é fundamental porque, no BA, a probabilidade de realizar uma busca local está inversamente relacionada ao volume e diretamente relacionada ao pulso: quanto maior a taxa de pulso e menor o volume, menor a frequência da busca local, o que pode comprometer a capacidade do algoritmo de refinar boas soluções encontradas. Por outro lado, uma frequência de busca local excessiva pode reduzir a diversidade da população, levando a uma estagnação em ótimos locais. Assim, o ajuste desses parâmetros é essencial para alcançar um bom equilíbrio entre exploração do espaço de busca e exploração intensiva em regiões promissoras. Como apresentado na Tabela 4, foram testadas três combinações distintas desses parâmetros, mantendo-se constante o intervalo padrão de frequência em [0,1; 1,0], a fim de isolar os efeitos específicos do pulso e do volume sobre o desempenho do algoritmo.

Tabela 4: Sequência de testes para melhor combinação de pulso e volume.

Teste	Pulso	Volume
1	0,1	0,9
2	0,4	0,7
3	0,7	0,4

Fonte: Elaborada pelo autor (2025).

A partir da melhor combinação previamente identificada de pulso e volume, foram realizados novos experimentos com o objetivo de avaliar o impacto do intervalo de frequência sobre o desempenho do algoritmo. No BA, a frequência controla a velocidade de movimento dos morcegos no espaço de busca, influenciando diretamente sua capacidade de explorar regiões distantes ou refinar áreas promissoras. Valores mais altos de frequência tendem a promover movimentos maiores, favorecendo a exploração global do espaço de busca, enquanto frequências mais baixas conduzem a deslocamentos menores, facilitando a exploração local e o refinamento de soluções. Portanto, a escolha adequada do intervalo de frequência mínima e máxima é crucial para manter um equilíbrio eficaz entre exploração e intensificação, dois fatores essenciais para evitar a estagnação em ótimos locais e garantir a convergência para boas soluções. A Tabela 5 apresenta os diferentes

intervalos de frequência testados, os quais foram avaliados em busca de um desempenho mais robusto e consistente do algoritmo.

Tabela 5: Sequência de testes para melhor intervalo de frequência.

Teste	Frequência mínima	Frequência Máxima
4	0,01	0,05
5	0,05	0,1
6	0,1	0,7
7	0,7	1,3
8	1,3	1,9

Fonte: Elaborada pelo autor (2025).

Dada a quantidade significativa de parâmetros envolvidos no algoritmo e a complexidade inerente à busca pela melhor combinação entre todos eles, foi necessário fixar alguns valores de forma empírica, mostrados na Tabela 6, com base em considerações práticas e resultados obtidos em testes preliminares. O critério de parada foi definido como 20 iterações, enquanto o número total de morcegos foi estabelecido em 16, distribuídos em uma única subpopulação. Os pesos atribuídos à função objetivo foram fixados em 1, 0,3 e 0,2, respectivamente, refletindo a importância relativa de cada componente conforme observado nos resultados de experimentos iniciais. Dessa forma, a função objetivo utilizada nos testes deste trabalho pode ser vista como:

$$Objective_i^t = \frac{1 \times Acc_i^t}{1 + (0,3 \times FLOPS_{ratio}) + (0,2 \times Params_{ratio})}. \tag{18}$$

Além disso, impôs-se um limite máximo de FLOPs igual a 2, o que significa que o algoritmo aceita soluções com até o dobro do custo computacional em relação ao *baseline*, permitindo maior flexibilidade na busca por modelos mais precisos sem comprometer a eficiência computacional. Por fim, como o protocolo de testes foi aplicado individualmente para cada arquitetura, obteve-se um total de 16 testes, gerando 16 novos modelos promissores, 8 para cada rede utilizada.

6.1.2 Protocolo de Treinamento

Após a etapa de otimização, os modelos gerados foram submetidos a um treinamento completo utilizando validação cruzada do tipo 3-*fold*, conforme descrito na Seção 4.2. Para

Tabela 6: Parâmetros do otimizador fixados.

Parâmetros	Valores
Iterações	20
Sub-populações	1
Morcegos	16
FLOPs máximo	2
B_1	1
B_2	0,3
B_3	0,2

Fonte: Elaborada pelo autor (2025).

assegurar uma comparação justa entre as arquiteturas otimizadas e seus respectivos *baselines*, adotou-se um protocolo de treinamento unificado. Todas as redes foram treinadas por 120 épocas, com um tamanho de lote *batch size* de 16 e uma taxa de aprendizado fixa de 10^{-5}. No caso da arquitetura VGGNet, foi aplicada uma taxa de abandono *dropout* de 20% entre cada camada totalmente conectada da rede, com o objetivo de reduzir o risco de (*overfitting*). O *dropout* atua desligando aleatoriamente uma fração dos neurônios durante o treinamento, forçando a rede a não depender excessivamente de unidades específicas, o que contribui para melhor generalização do modelo em dados não vistos. A resolução de entrada das imagens foi padronizada em 224×224 para todas as arquiteturas, garantindo consistência entre os experimentos. A Tabela 7 apresenta um resumo dos parâmetros de treinamento adotados.

Tabela 7: Parâmetros do otimizador fixados.

Parâmetros	VGGNet	DenseNet
Batch size	16	16
Épocas	120	120
Taxa de Aprendizado	10^{-5}	10^{-5}
Taxa de *Dropout*	0,2	-
Resolução	224	224

Fonte: Elaborada pelo autor (2025).

Como o objetivo principal deste trabalho é avaliar a influência da estrutura e das dimensões de redes convolucionais na qualidade da classificação de imagens, optou-se por não aplicar técnicas de modificação da base de dados, como aumento de dados (*data augmentation*) ou métodos de balanceamento. Dessa forma, evita-se a introdução de variáveis externas que poderiam mascarar os efeitos da arquitetura da rede sobre o desempenho.

Os experimentos foram conduzidos utilizando a função de perda de entropia cruzada, com o otimizador SGD configurado com momento igual a 0,9, decaimento de pesos de 10^{-4} e ativação do método *Nesterov*, o qual garante uma convergência mais rápida e estável do modelo. Além disso, empregou-se o algoritmo de escalonamento da taxa de aprendizado *ReduceLROnPlateau*, fornecido pela biblioteca *PyTorch*, com o objetivo de mitigar o sobreajuste. Esse algoritmo foi configurado para monitorar a acurácia de validação, reduzindo a taxa de aprendizado em um fator de 0,9 sempre que não fosse observada melhoria após 40 épocas consecutivas. Essa estratégia permite ajustes dinâmicos da taxa de aprendizado durante o treinamento, favorecendo uma convergência mais estável e eficaz. É válido destacar que os parâmetros de treinamento são os mesmos para a etapa de otimização, com a diferença de que os modelos são treinados no conjunto de dados *Optimization Set* durante a execução do BA.

Para avaliar numericamente o desempenho dos modelos no problema de classificação de raças bovinas, foi utilizada a métrica de acurácia, uma das medidas mais comuns em tarefas de classificação. A acurácia representa a proporção de previsões corretas em relação ao total de amostras avaliadas, sendo calculada a partir da razão entre as predições corretas e o total de amostras. Por se tratar de uma classificação binária, uma vez que há apenas duas classes no problema, as predições corretas são calculadas a partir da soma entre os verdadeiros positivos (*true positive* – TP) e verdadeiros negativos (*true negative* – TN). Analogamente, o total de predições é calculado pela soma entre as predições corretas e as predições erradas, as quais são dadas pela soma entre os falsos positivos (*false positive* – FP) e os falsos negativos (*false negative* – FN). Essa métrica, cuja fórmula é:

$$\text{Acurácia} = \frac{TP + TN}{TP + TN + FP + FN}, \tag{19}$$

foi utilizada tanto para medir o desempenho individual dos modelos, quanto para compor a função objetivo do BA e o valor considerado foi o obtido a partir do desempenho no conjunto de validação após a última época de treinamento.

6.2 Treinamento dos *Baselines*

Antes da execução dos experimentos de otimização, foi realizado o treinamento e avaliação dos modelos *baseline*, os quais servem como ponto de referência para medir os ganhos obtidos pelas arquiteturas otimizadas. Entre os modelos avaliados, a VGGNet foi utilizada em sua configuração completa, com 100% de profundidade (11 camadas) e 100% de largura, apresentando blocos convolucionais com 64, 128, 256, 512 e 512 filtros. Essa configuração resultou em um custo computacional de 320,13 Giga FLOPs e aproximadamente 45,12 milhões de parâmetros treináveis. Por outro lado, a DenseNet, também utilizada em sua configuração original com 121 camadas de profundidade e fator de crescimento $k = 32$, demonstrou uma eficiência computacional superior, exigindo apenas 126,27 *Giga* FLOPs e 11,99 milhões de parâmetros. Essas diferenças, mostradas na Tabela 8, têm implicações diretas no desempenho e na viabilidade prática dos modelos. O maior número de FLOPs da VGGNet representa um custo computacional significativamente mais alto, o que impacta o tempo de treinamento, o consumo de energia e a capacidade de implantação do modelo em ambientes com recursos limitados. Durante o treinamento, a rede demandava aproximadamente 190 segundos por época, por exemplo, em comparação à DenseNet, que demandava aproximadamente 170 segundos. Além disso, o elevado número de parâmetros aumenta a demanda por memória e pode tornar o modelo mais suscetível a sobreajuste, especialmente em conjuntos de dados com tamanho limitado.

Tabela 8: Dimensões e custo computacional dos *baselines*.

Modelo	Profundidade	Largura	FLOPs (G)	Parâmetros (M)
VGGNet	1,0	1,0	320,13	45,12
DenseNet	1,0	1,0	126,27	11,99

Fonte: Elaborada pelo autor (2025).

Apesar de apresentar uma eficiência computacional superior em relação à VGGNet, a DenseNet obteve desempenho inferior em termos de acurácia na maioria dos cenários avaliados, os quais são exibidos na Tabela 9. A única exceção observada foi na acurácia de teste da raça Holandês, onde ela atingiu 90,38%, superando ligeiramente sua concorrente, que obteve 89,90%. A VGGNet demonstrou desempenho consistentemente superior nos demais valores avaliados, especialmente na classificação da raça minoritária

Girolando. Nessa classe, ela superou a DenseNet em cerca de 8% no conjunto de validação e 5% no conjunto de testes, evidenciando uma maior capacidade de generalização para a classe com menor representatividade.

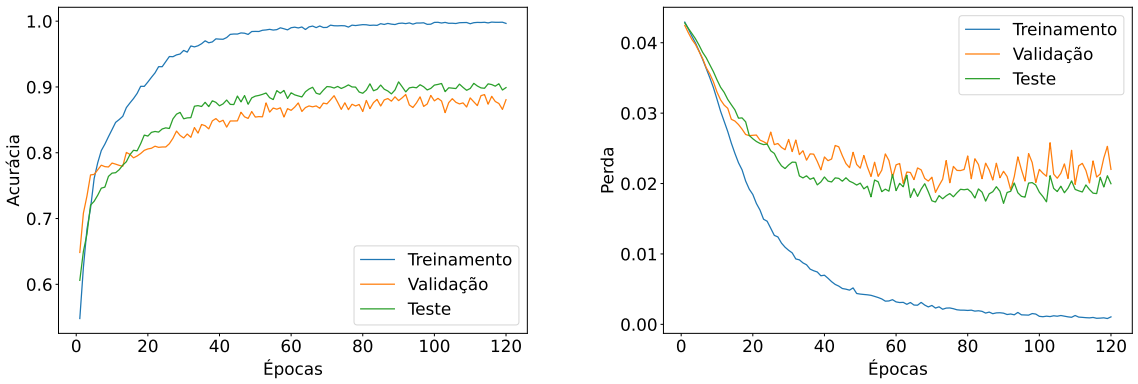
Tabela 9: Resultados de acurácia geral e das classes para os *baselines*.

Modelo	Geral (%)		Girolando (%)		Holandês (%)	
	Val.	Teste	Val.	Teste	Val.	Teste
VGGNet	88,06	89,91	82,45	89,73	92,07	89,90
DenseNet	83,17	87,47	74,17	84,05	90,05	90,38

Fonte: Elaborada pelo autor (2025).

Além disso, os gráficos de treinamento da VGGNet, representados na Figura 15a, mostram um comportamento estável da acurácia, considerando o resultado médio dos três *splits* da validação cruzada. Esse padrão sugere que o modelo apresenta um processo de aprendizado consistente, com bom nível de generalização em diferentes partições do conjunto de dados, mantendo um desempenho equilibrado entre os conjuntos e minimizando o sobreajuste. Por outro lado, os gráficos da função de perda, apresentados na Figura 15b, revelam oscilações na perda nos conjuntos de validação e teste a partir da trigésima época. Isso pode estar relacionado ao fato de que, a partir deste ponto, o modelo pode já ter atingido sua estabilidade no treinamento, tornando o treinamento nas épocas subsequentes mais sensível a sobreajuste.

Figura 15: Resultados gerais do treinamento da VGGNet *baseline*.



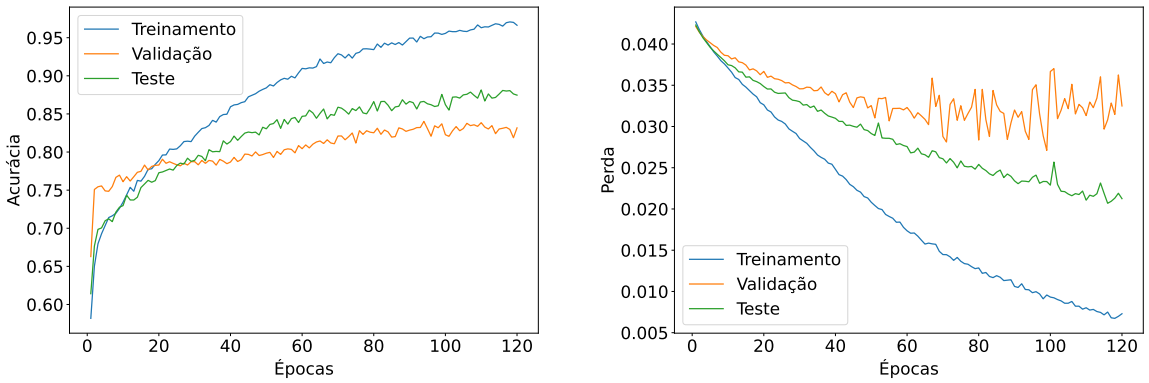
(a) Acurácia (b) Perda

Fonte: Elaborada pelo autor (2025).

A análise dos gráficos de treinamento da DenseNet, apresentados na Figura 16a,

revela um comportamento igualmente estável na evolução da acurácia para os conjuntos de treinamento e teste, porém com o conjunto de validação tendo um desempenho insatisfatório. Observa-se que sua curva sofre um aumento brusco nas primeiras épocas, atingindo rapidamente um platô, o que sugere que o modelo encontrou dificuldades no processo de aprendizado para algumas partições do conjunto de validação no 3-*fold*. Aliado a isso, a Figura 16b evidencia uma alta instabilidade na função de perda de validação, com as oscilações se acentuando ao longo das épocas. Essas flutuações podem estar associadas a dificuldades de generalização do modelo, indicando que a DenseNet demonstrou maior sensibilidade às variações entre os *splits* da validação cruzada nesse conjunto.

Figura 16: Resultados gerais do treinamento da DenseNet *baseline*.



(a) Acurácia (b) Perda

Fonte: Elaborada pelo autor (2025).

Esses resultados evidenciam que, apesar da importância da eficiência computacional, a estrutura interna dos modelos desempenha um papel fundamental na capacidade de aprendizado e generalização. Portanto, destaca-se a necessidade de otimização sistemática dessas dimensões, a fim de alcançar um melhor equilíbrio entre desempenho e custo computacional. Esse fato reforça a relevância da etapa de otimização proposta neste trabalho, que busca ajustar automaticamente esses parâmetros estruturais para encontrar arquiteturas mais acuradas e eficientes.

6.3 Otimização das Arquiteturas *Baseline*

A primeira etapa dos testes de otimização teve como objetivo identificar a melhor combinação entre os parâmetros de pulso e volume do algoritmo baseado em morcegos para cada uma das redes avaliadas. Os resultados, exibidos entre linhas 1 e 3 das Tabelas 10 e 11 demonstraram que, para a VGGNet, a combinação que proporcionou o melhor desempenho da função objetivo, de 0,6122 no Teste 1, foi com pulso igual a 0,1 e volume igual a 0,9, favorecendo uma maior intensidade de busca local e uma movimentação mais exploratória no início do processo. Já para a DenseNet, o melhor resultado, de 0,6388 do Teste 2, foi obtido com pulso igual a 0,4 e volume igual a 0,7, sugerindo que essa arquitetura se beneficia de uma frequência de busca local mais elevada e um volume ligeiramente mais moderado, promovendo um equilíbrio mais eficaz entre exploração e aprimoramento no espaço de soluções.

Tabela 10: Resultados da Otimização da VGGNet.

Teste	Objetivo	Profundidade	Largura	FLOPs (G)	Parâmetros (M)
1	0,6152	0,58	0,20	5,89	2,92
2	0,6077	1,26	0,20	15,46	1,41
3	0,6122	1,20	0,20	15,46	1,41
4	0,5782	0,72	0,45	38,82	11,01
5	0,5785	0,64	0,30	17,86	4,96
6	0,6043	1,26	0,26	26,74	2,47
7	0,6276	1,26	0,20	14,46	1,41
8	0,6359	0,50	0,20	3,87	2,92

Fonte: Elaborada pelo autor (2025).

Aliado a isso, observou-se que ambas as redes apresentaram uma tendência consistente de redução de suas dimensões estruturais, tanto em profundidade quanto em largura. Essa compactação estrutural resultou em uma diminuição significativa do custo computacional comparado aos respecitvos *baselines* (Tabela 8) com queda dos valores de FLOPs e no número de parâmetros treináveis. Para a VGGNet, o melhor modelo encontrado apresentava apenas 58% da profundidade e 20% da largura da configuração original, o que levou a uma redução expressiva de 320,13 Giga FLOPs para apenas 5,89 Giga FLOPs, e de 45,12 milhões para 2,92 milhões de parâmetros. No caso da DenseNet, o modelo otimizado preservou 83% da profundidade original e 71% da largura, resultando

em uma redução de 126,27 Giga FLOPs para 38,77 Giga FLOPs e de 11,99 milhões para 3,63 milhões de parâmetros.

Tabela 11: Resultados da Otimização da DenseNet.

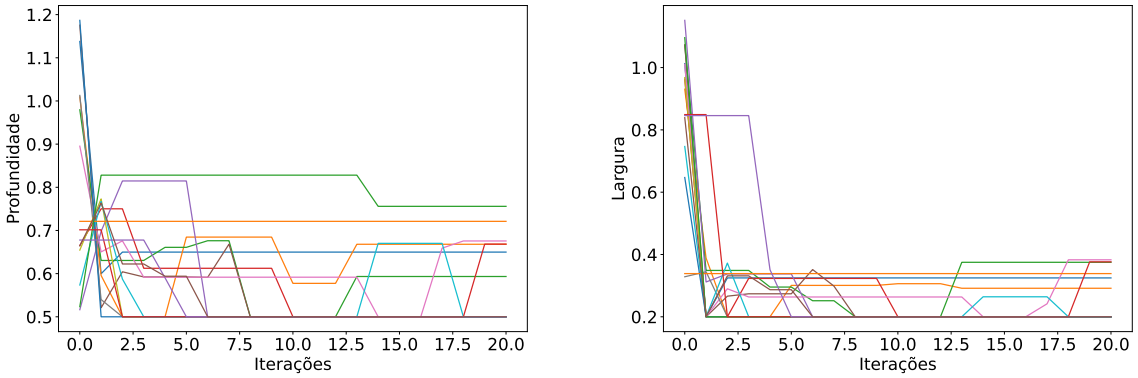
Teste	Objetivo	Profundidade	Largura	FLOPs (G)	Parâmetros (M)
1	0,6035	0,89	0,69	43,55	4,00
2	0,6388	0,83	0,71	38,77	3,63
3	0,6374	0,92	0,36	13,43	1,2
4	0,6501	1,01	0,37	16,14	1,45
5	0,6486	0,99	0,52	28,11	2,67
6	0,6873	0,24	1,26	23,25	1,36
7	0,6769	0,87	0,51	25,33	2,19
8	0,6311	0,80	0,44	16,26	1,44

Fonte: Elaborada pelo autor (2025).

Com a melhor combinação de pulso e volume já definida, os cinco testes subsequentes, Testes 4 a 8 nas Tabelas 10 e 11, foram direcionados à busca do melhor intervalo de frequência para cada arquitetura, visando refinar ainda mais o desempenho do algoritmo. Para a VGGNet, o melhor resultado da função objetivo foi alcançado no Teste 8, com valor de 0,6359. Nesse cenário, o modelo gerado possuía 50% da profundidade e 20% da largura da versão original, resultando em um custo computacional de apenas 3,87 Giga FLOPs e 2,92 milhões de parâmetros. No entanto, a análise dos gráficos de evolução dos morcegos, presentes na Figura 17, revela que o intervalo de frequência adotado nesse teste, [1,3; 1,9], por ser relativamente alto, ocasionou uma convergência muito prematura do algoritmo. Isso pode limitar a capacidade de exploração do espaço de busca, levando o BA a se fixar rapidamente em ótimos locais subótimos. Por outro lado, intervalos de frequência muito baixos, como [0,01; 0,05] no Teste 4, mostraram-se igualmente problemáticos, pois promoveram uma dispersão excessiva dos agentes, com cada um seguindo trajetórias muito distintas, o que dificultou a convergência do algoritmo, como mostrado na Figura 18

Para a DenseNet, o melhor resultado foi obtido no Teste 6, que utilizou um intervalo de frequência intermediário de [0,1; 0,7], alcançando um valor de 0,6873 na função objetivo. Nessa configuração, o modelo gerado apresentou uma redução expressiva da profundidade, mantendo apenas 24% da profundidade original, enquanto a largura foi expandida para 126%, o que indica uma redistribuição do poder representacional da rede.

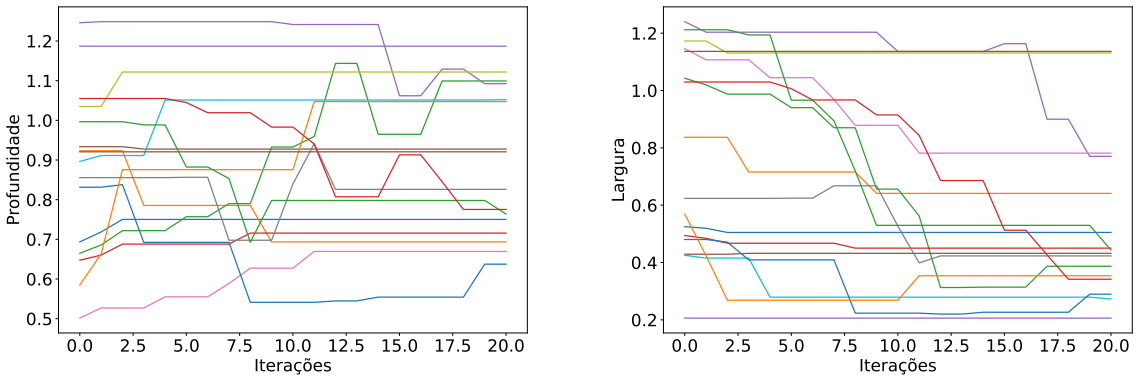
Figura 17: Evolução de cada morcego durante o Teste 8 com a VGGNet.



(a) Profundidade (b) Largura

Fonte: Elaborada pelo autor (2025).

Figura 18: Evolução de cada morcego durante o Teste 4 com a VGGNet.



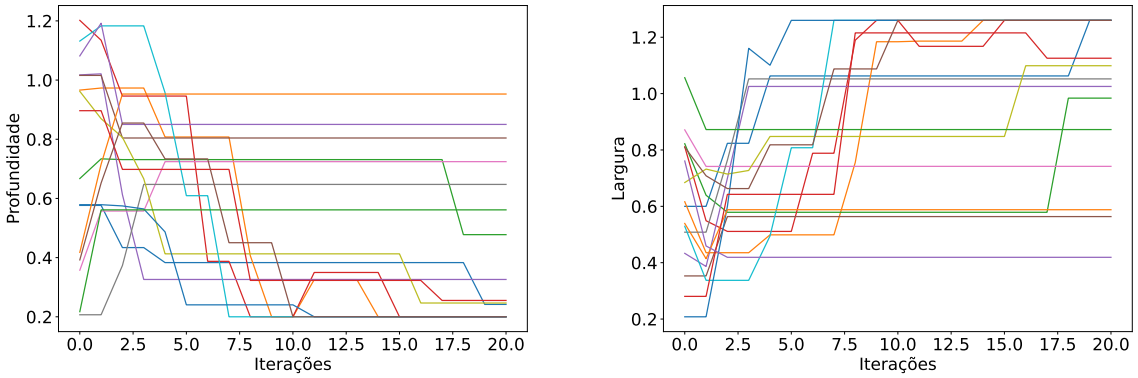
(a) Profundidade (b) Largura

Fonte: Elaborada pelo autor (2025).

Mesmo com essa expansão na largura, o modelo resultante mostrou-se bastante eficiente, com um consumo de apenas 23,25 Giga FLOPs e 1,36 milhões de parâmetros, valores significativamente menores que os da arquitetura base. A escolha de um intervalo de frequência intermediário demonstrou ser eficaz ao proporcionar uma convergência relativamente estável, como visto na Figura 19, conseguindo equilibrar de forma razoável a exploração do espaço de busca, com um refinamento dessas soluções.

Tendo em vista os resultados obtidos, entende-se a necessidade de explorar o impacto de outros parâmetros do algoritmo em seu funcionamento. O critério de parada escolhido neste trabalho, por exemplo, também pode ser um fator determinante, uma vez que um limite de iterações muito baixo pode prejudicar a convergência dos morcegos.

Figura 19: Evolução de cada morcego durante o Teste 6 com a DenseNet.



(a) Profundidade (b) Largura

Fonte: Elaborada pelo autor (2025).

Outro ponto a ser observado é que a construção de uma única subpopulação pode não ser o ideal, uma vez que todos os morcegos estarão limitados aos parâmetros dessa população. Neste caso, múltiplos grupos de indivíduos, com diferentes combinações de parâmetros, são capazes de explorar melhor o espaço de busca, além da possibilidade de trocarem informações entre eles, através de mecanismos de *crossover*, gerando uma variabilidade maior de soluções.

6.4 Treinamento dos Modelos Otimizados

Após a etapa de otimização, cada modelo gerado passou por um treinamento completo no 3-*fold* estruturado. A Tabela 12 mostra os resultados de acurácia geral da validação cruzada para a VGGNet. Nela, é possível observar que parte dos modelos gerados foi capaz de superar os resultados do *baseline*, com destaque para as redes VGGNet do Teste 1 e do Teste 4, os quais apresentaram desempenhos superiores tanto na acurácia geral de validação quanto na de teste. Além dessas, os Testes 5 e 8 também apresentaram melhorias em alguns dos contextos analisados. Porém, no caso do Teste 5, por exemplo, essa melhoria é alcançada apenas na classe Holandês, em detrimento da piora no desempenho da classe Girolando.

Os gráficos mostrados na Figura 20 evidenciam uma melhoria na estabilidade do treinamento, tanto na acurácia quanto na perda para a rede do Teste 1 em relação ao

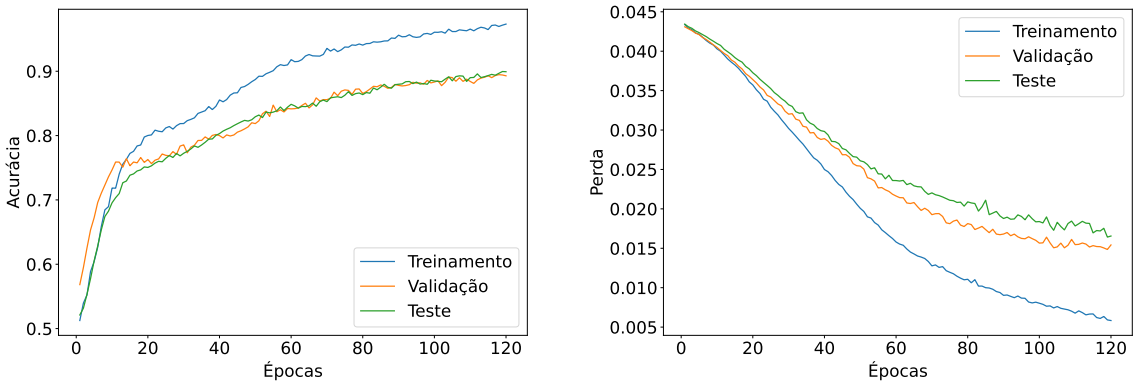
Tabela 12: Resultados da acurácia geral e da acurácia das classes para os testes de otimização com a VGGNet. Resultados superiores ao *baseline* estão destacados em negrito.

Teste	Geral (%)		Girolando (%)		Holandês (%)	
	Val.	Teste	Val.	Teste	Val.	Teste
Baseline	88,06	89,91	82,45	89,73	92,07	89,90
1	**89,29**	**89,93**	**86,38**	88,28	91,20	**91,30**
2	79,92	85,30	71,07	81,95	86,29	88,18
3	81,66	86,18	72,58	82,56	88,25	89,37
4	**88,32**	**90,48**	81,12	88,27	**93,38**	**92,33**
5	86,47	86,94	78,34	83,60	**92,54**	**89,96**
6	82,08	85,81	72,75	82,38	88,48	88,71
7	80,15	83,71	74,24	83,37	84,87	83,72
8	**88,19**	89,79	**85,50**	87,26	89,84	**91,81**

Fonte: Elaborada pelo autor (2025).

baseline (Figura 15). Os gráficos do modelo do Teste 4 (Figura 21) também se mostraram promissores, porém a instabilidade na perda da validação volta a aparecer em épocas posteriores do treinamento. Isso pode estar relacionado ao fato de que, apesar de essa rede superar o *baseline* nas acurácias gerais, essa melhoria é proveniente de um melhor desempenho de classificação da classe majoritária (Holandês), fazendo com que, à medida que novas épocas sejam realizadas, a classe minoritária (Girolando) possa apresentar perda de desempenho.

Figura 20: Resultados gerais do treinamento da arquitetura VGGNet do Teste 1.



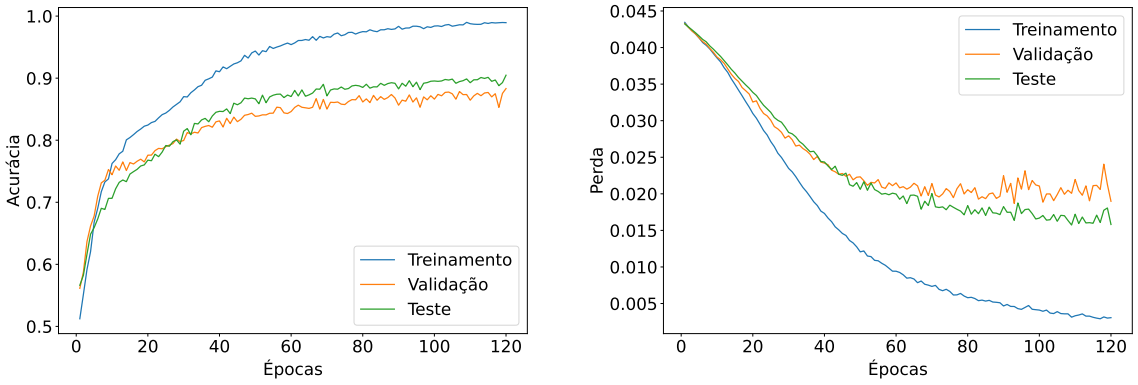
(a) Acurácia (b) Perda

Fonte: Elaborada pelo autor (2025).

Os demais experimentos com a VGGNet (Testes 2, 3, 6 e 7) apresentaram resultados insatisfatórios, pois, apesar do significativo aumento na eficiência dos modelos, eles não foram capazes de superar nenhum dos resultados de acurácia do *baseline*. Isso pode

estar relacionado ao fato de que há uma discrepância muito grande entre as dimensões dessas redes, com a profundidade variando de 120% a 126% e largura variando de 20% a 26%. Neste caso, os modelos construídos (muito profundos e estreitos) não apresentam o equilíbrio necessário das dimensões para esse problema de classificação. Além disso, é válido destacar que todos os modelos apresentaram valores de largura próximos ao limite inferior do espaço de busca, com os melhores modelos possuindo profundidades semelhantemente menores, variando de 50% a 72%.

Figura 21: Resultados gerais do treinamento da arquitetura VGGNet do Teste 4.



(a) Acurácia (b) Perda

Fonte: Elaborada pelo autor (2025).

Para os resultados de treinamento das redes baseadas na DenseNet, a Tabela 13 mostra que um conjunto maior de 7 modelos demonstrou melhorias em relação ao *baseline*, contra 4 da VGGNet, com destaque para DenseNet do Teste 2 e DenseNet do Teste 6, que foram capazes de aprimorar a acurácia geral de validação. Apesar dessa vantagem de quantidade em relação à VGGNet, as melhorias ocorridas nos testes atuais são pontuais, com foco total na classe minoritária (Girolando) e uma queda significativa das acurácias da classe Holandês. Essa melhoria pode estar relacionada à natureza da própria arquitetura, a qual, ao concatenar os filtros das camadas anteriores, é capaz de aprender melhor as diferenças entre as classes, trazendo um equilíbrio melhor entre elas. Porém, observa-se um cenário em que as alterações nas dimensões influenciaram positivamente apenas uma das classes, o que abre a possibilidade para novos estudos no futuro.

Observando os gráficos de treinamento do modelo DenseNet do Teste 2, pode-se observar que, apesar de resultados melhores de acurácia de validação, o treinamento se

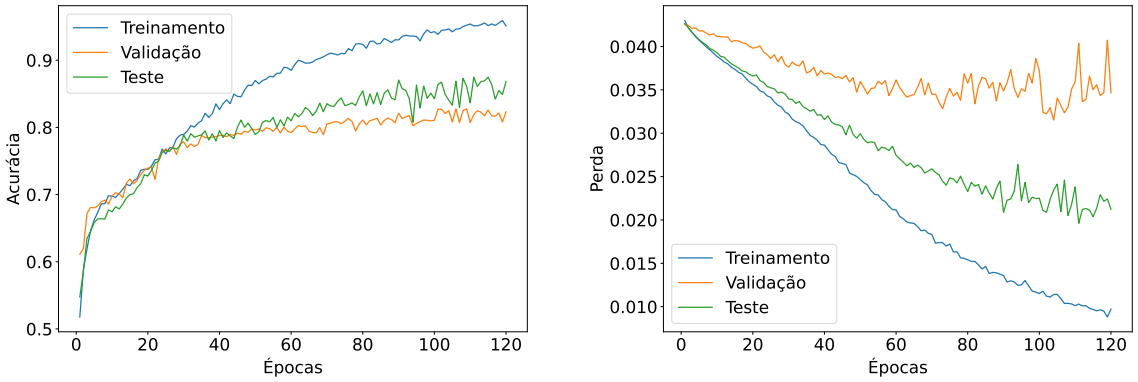
Tabela 13: Resultados da acurácia geral e da acurácia das classes para os testes de otimização com a DenseNet. Resultados superiores ao *baseline* estão destacados em negrito.

Teste	Geral (%)		Girolando (%)		Holandês (%)	
	Val.	Teste	Val.	Teste	Val.	Teste
Baseline	83,17	87,47	74,17	84,05	90,05	90,38
1	82,59	86,20	**78,12**	**85,18**	86,27	86,81
2	**83,70**	86,60	**76,01**	**84,29**	89,61	88,70
3	80,94	85,39	69,77	81,53	89,53	88,83
4	82,87	86,68	**77,00**	**86,41**	87,05	86,87
5	82,54	85,98	**75,30**	**84,15**	87,96	87,55
6	**83,17**	87,05	**74,88**	**84,60**	89,68	89,14
7	82,23	86,15	72,87	**84,49**	89,35	87,56
8	82,25	87,03	**74,70**	**85,81**	88,22	87,90

Fonte: Elaborada pelo autor (2025).

mostrou mais instável que a rede original, com curvas de acurácia de validação e teste (Figura 22), apresentando maiores oscilações que o *baseline* (Figura 16) principalmente no conjunto de teste, bem como curvas de perda de validação com tendência de crescimento após um determinado número de épocas. Apesar disso, houve uma ligeira melhora na perda de teste, com as últimas épocas atingindo valores abaixo de 0,20.

Figura 22: Resultados gerais do treinamento da arquitetura DenseNet do Teste 2.



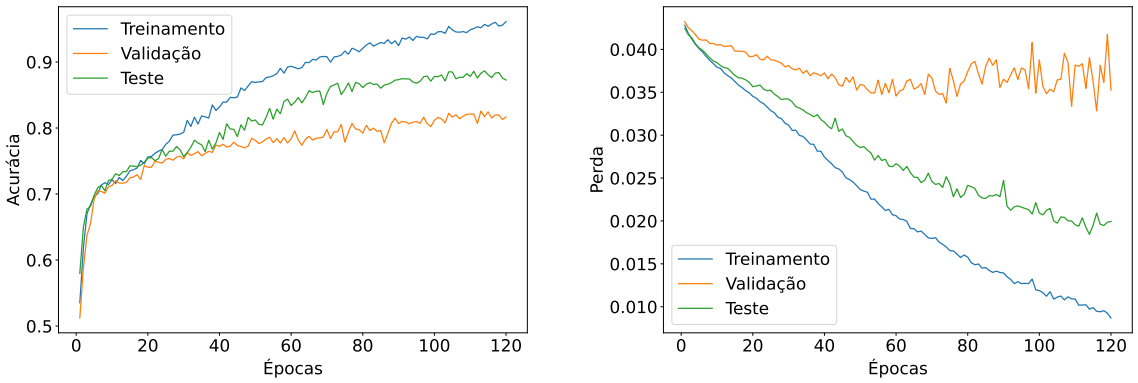
(a) Acurácia (b) Perda

Fonte: Elaborada pelo autor (2025).

O mesmo comportamento é observado na DenseNet do Teste 6, onde a curva de perda de validação (Figura 23) apesar de se manter estável durante as primeiras 40 épocas, apresenta oscilações crescentes ao longo do restante do treinamento. Em ambos os modelos, observa-se que, apesar dessa instabilidade nos conjuntos de validação e teste, a rede ainda não atingiu sua convergência de treinamento, mostrando que ela ainda tem

capacidade de aprendizado dos dados de treinamento caso novas épocas sejam adicionadas. Esse comportamento pode representar a ocorrência de sobreajuste, onde a rede está se adaptando demais aos dados de treinamento e aumentando a perda dos outros conjuntos ao longo das épocas. Por outro lado, tendo em vista que as curvas no conjunto de teste evoluem de forma consistente, o comportamento observado pode estar relacionado apenas à dificuldade do modelo em se adaptar à configuração da base no conjunto de validação.

Figura 23: Resultados gerais do treinamento da arquitetura DenseNet do Teste 6.



(a) Acurácia (b) Perda

Fonte: Elaborada pelo autor (2025).

Os demais modelos treinados (DenseNet dos Testes 1, 3, 4, 5, 7 e 8), apesar de não atingirem resultados satisfatórios, os valores de acurácia não foram tão inferiores ao *baseline* em comparação com os testes com a VGGNet. Enquanto os menores valores de validação e teste gerais da DenseNet tiveram uma diferença de, respectivamente, 2,23% e 2,08% em relação à rede base, os menores valores da VGGNet obtiveram uma diferença de 8,14% e 6,20%. Esse comportamento demonstra que a VGGNet possui uma sensibilidade maior para o escalonamento da profundidade e largura no problema de classificação de raças de bovinos em relação à DenseNet.

Comparando os resultados obtidos entre os melhores modelos construídos (Tabela 14) com os modelos base estabelecidos neste trabalho, conclui-se que o algoritmo de otimização proposto foi capaz de escalonar de forma satisfatória e eficiente a profundidade e a largura das redes, encontrando arquiteturas eficientes e eficazes para o problema de classificação abordado. Dentre elas, a que melhor desempenhou foi a VGGNet do Teste 1, a qual apresentou os melhores valores de acurácia geral de validação e teste,

em relação aos resultados de ambas as arquiteturas, bem como demandou a execução do menor número de operações de ponto flutuante dentre todos os experimentos, com uma diminuição de 98,2% do número de FLOPs em relação à VGGNet *baseline* e 95,4% em relação à DenseNet *baseline*. Além disso, ela conseguiu reduzir o gasto de parâmetros em aproximadamente 93% e 76% comparado aos modelos base, respectivamente.

Tabela 14: Comparação entre os resultados das melhores arquiteturas construídas.

Modelo	Teste	Acurácia Geral (%)		Custo		
		Val.	Teste	FLOPs (G)	Parâmetros (M)	Tempo por Época (s)
VGGNet	*Baseline*	88,06	89,91	320,13	45,12	152,92
VGGNet	1	89,29	89,93	5,89	2,92	139,75
VGGNet	4	88,32	90,48	38,82	11,01	143,31
DenseNet	*Baseline*	83,17	87,47	126,27	11,99	177,68
DenseNet	2	83,70	86,60	38,77	3,63	160,24
DenseNet	6	83,17	87,05	23,25	1,36	144,49

Fonte: Elaborada pelo autor (2025).

Apesar do método proposto ter-se mostrado eficaz na otimização das arquiteturas convolucionais determinadas, algumas dificuldades foram observadas durante a construção do trabalho. O espaço de busca das redes, por exemplo, foi um fator limitante das soluções. Enquanto a DenseNet base possuía um número significativamente alto de camadas (121), o que faz com que uma mínima variação no escalonador de profundidade surta efeito nessa quantidade, a VGGNet, com apenas 11 camadas, necessitava de uma alteração muito alta no escalonador para que a quantidade de camadas realmente se alterasse. Além disso, o algoritmo de otimização baseado em morcegos não é isento do fator de aleatoriedade, uma vez que os morcegos da população inicial são inicializados em posições aleatórias do espaço de busca. Se o algoritmo se iniciar com uma grande variedade de morcegos, o espaço de busca será melhor explorado; caso contrário, pode haver uma convergência prematura dos indivíduos. Elaborar testes com múltiplas populações e aplicação de técnicas de *crossover* pode contribuir para a amenização desse problema.

Outro aspecto observado foi o tempo de execução dos testes de otimização. Considerando que o número de treinamentos realizados durante a execução do algoritmo é diretamente proporcional ao número de iterações, à quantidade de morcegos e ao número de épocas de treinamento da rede, quanto maiores forem esses valores, mais tempo será gasto para encontrar um modelo otimizado. Nos experimentos deste trabalho, por exem-

plo, o tempo médio de execução do BA foi de 3 horas utilizando a amostragem da base de dados total, definida na Tabela 1 (*Optimization Set*). No cenário ideal, onde a base inteira é utilizada para a etapa de otimização, o gasto de tempo seria significativamente maior que o observado nos testes.

7 Considerações Finais

A aplicação de métodos de Visão Computacional e aprendizado profundo em áreas relacionadas à pecuária tem se tornado um grande objeto de estudo e interesse nos últimos anos, uma vez que essa utilização pode reduzir a necessidade de trabalhos manuais e aumentar a qualidade da produção. Neste sentido, destaca-se que construir o modelo de rede convolucional para cada problema é uma tarefa complicada e manual, com a escolha incorreta da arquitetura e sua estrutura sendo um fator drástico na qualidade das soluções e eficiência do método.

A partir disso, o presente trabalho propôs a criação de um método automatizado de otimização da estrutura interna de uma rede convolucional (profundidade e largura), baseado em um algoritmo bioinspirado em uma população de morcegos. Através dele, este estudo foi capaz de encontrar redes mais eficientes que as já estabelecidas na literatura e com resultados de classificação superiores. Dadas as duas arquiteturas escolhidas, a VGGNet obteve melhor resultado, apresentando um modelo final significativamente menos custoso que o original e com resultados superiores de acurácia geral nos conjuntos de validação e teste. Essa rede conseguiu reduzir o custo de FLOPs e número de parâmetros em mais de 90%. Diferentemente, a sua concorrente DenseNet, apesar de apresentar uma melhoria da classificação da classe minoritária, não foi capaz de superar de maneira satisfatória a sua arquitetura base, a qual já possuía resultados inferiores à VGGNet.

Durante a etapa de otimização dos testes, observou-se como os hiperparâmetros do BA afetam o comportamento dos morcegos. Enquanto intervalos de frequência muito baixos impedem os morcegos de evoluírem de forma eficaz, intervalos de frequência muito altos podem gerar uma convergência prematura da população, prejudicando a exploração do espaço de busca. Outro aspecto envolve a escolha dos parâmetros responsáveis pela busca local. Equilibrar o pulso e volume da forma correta torna mais eficaz o aprimoramento das soluções correntes.

De forma geral, os experimentos realizados evidenciam que o método de otimização proposto é capaz de criar arquiteturas eficientes e personalizadas para o problema

de classificação binária de raças de bovinos, com possibilidade de expansão da abordagem para outros problemas da área de Visão Computacional, como detecção de objetos e segmentação. Apesar disso, alguns fatores devem ser considerados em experimentos futuros, como as diferenças das arquiteturas utilizadas, as quais podem afetar a exploração do espaço de soluções. Uma rede base rasa torna necessário uma variação muito drástica no escalonador de profundidade para que de fato haja alteração no número de camadas, enquanto uma rede muito profunda é significativamente mais sensível a variações no valor do escalonador. Analogamente, dada a necessidade de executar um número significativo de treinamentos durante a otimização, o tempo de execução será igualmente elevado, prejudicando a eficiência dessa etapa da tarefa.

Portanto, avalia-se este cenário como uma grande oportunidade para realização de novos estudos. Propõe-se como trabalho futuro a possibilidade de testar novas arquiteturas além das já utilizadas, com o objetivo de entender como as diferentes propostas estruturais afetam a precisão na solução de um problema de Visão Computacional. Não obstante disso, entende-se que é importante realizar comparações entre o algoritmo baseado em morcegos com outras meta-heurísticas, de forma a se avaliar qual deles se adequa melhor à tarefa de otimização estrutural de CNNs.

Bibliografia

ADHIKARI, B.; HUTTUNEN, H. Iterative bounding box annotation for object detection. In: IEEE. *2020 25th International Conference on Pattern Recognition (ICPR)*. [S.l.], 2021. p. 4040–4046.

BACANIN, N. et al. Dropout probability estimation in convolutional neural networks by the enhanced bat algorithm. In: IEEE. *2020 International Joint Conference on Neural Networks (IJCNN)*. [S.l.], 2020. p. 1–7.

BENAISSA, B. et al. Metaheuristic optimization algorithms: An overview. *HCMCOU Journal of Science–Advances in Computational Structures*, p. 33–61, 2024.

BEZERRA, E. Introdução à aprendizagem profunda. *Artigo–31º Simpósio Brasileiro de Banco de Dados–SBBD2016–Salvador*, 2016.

BOTTOU, L. Stochastic gradient descent tricks. In: *Neural networks: tricks of the trade: second edition*. [S.l.]: Springer, 2012. p. 421–436.

CHOLLET, F. Xception: Deep learning with depthwise separable convolutions. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2017. p. 1251–1258.

DAC, H. H. et al. Livestock identification using deep learning for traceability. *Sensors*, MDPI, v. 22, n. 21, p. 8256, 2022.

DORIGO, M. Optimization, learning and natural algorithms. *Ph. D. Thesis, Politecnico di Milano*, 1992.

FERREIRA, F. C.; SIQUEIRA, K. B.; PEREIRA, L. G. R. A pecuária leiteira de precisão sob a ótica econômica. *Cad. técn. Vet. Zoot.*, p. 137–145, 2015.

GARBACCIO, G. L.; PRATLONG, F. Perspectivas atuais da sustentabilidade: Cenários internacional e brasileiro. Instituto Brasileiro de Ensino, Desenvolvimento e Pesquisa, 2024.

GONÇALVES, C. B.; SOUZA, J. R.; FERNANDES, H. Cnn architecture optimization using bio-inspired algorithms for breast cancer detection in infrared images. *Computers in Biology and Medicine*, Elsevier, v. 142, p. 105205, 2022.

GONZALEZ, R. C. *Digital image processing*. [S.l.]: Pearson education india, 2009.

GWILLIAM, M.; FARRELL, R. Intelligent image collection: Building the optimal dataset. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. [S.l.: s.n.], 2020. p. 796–805.

HE, K. et al. Deep residual learning for image recognition. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2016. p. 770–778.

HOLLAND, J. H. Adaptation in natural and artificial systems. an introductory analysis with applications to biology, control and artificial intelligence. *Ann Arbor: University of Michigan Press*, 1975.

HUANG, G. et al. Densely connected convolutional networks. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2017. p. 4700–4708.

IONESCU, R. T. et al. How hard can it be? estimating the difficulty of visual search in an image. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2016. p. 2157–2166.

JARA, M.; SHETA, A.; ELASHMAWI, W. H. Cnn-based early diagnosis of breast cancer with variable image resolutions. *International Journal of Computer Applications*, v. 975, p. 8887, 2024.

KHEMASUWAN, D.; SORENSEN, J. S.; COLT, H. G. Artificial intelligence in pulmonary medicine: computer vision, predictive model and covid-19. *European respiratory review*, European Respiratory Society, v. 29, n. 157, 2020.

KHOSLA, A. et al. Novel dataset for fine-grained image categorization: Stanford dogs. In: *Proc. CVPR workshop on fine-grained visual categorization (FGVC)*. [S.l.: s.n.], 2011. v. 2, n. 1.

KRIZHEVSKY, A.; HINTON, G. et al. Learning multiple layers of features from tiny images. Toronto, ON, Canada, 2009.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, AcM New York, NY, USA, v. 60, n. 6, p. 84–90, 2017.

LECUN, Y. et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, Ieee, v. 86, n. 11, p. 2278–2324, 2002.

LI, X. Exploring the effect of depth and width of cnn models on binary classification of dogs and cats. *Applied and Computational Engineering*, v. 47, n. 1, p. 147–158, 2024.

LUO, C. et al. How does the data set affect cnn-based image classification performance? In: IEEE. *2018 5th international conference on systems and informatics (ICSAI)*. [S.l.], 2018. p. 361–366.

LUO, Z. et al. Mio-tcd: A new benchmark dataset for vehicle classification and localization. *IEEE Transactions on Image Processing*, IEEE, v. 27, n. 10, p. 5129–5141, 2018.

MARTINEZ, P.; AL-HUSSEIN, M.; AHMAD, R. A scientometric analysis and critical review of computer vision applications for construction. *Automation in Construction*, Elsevier, v. 107, p. 102947, 2019.

QIU, J. An analysis of model evaluation with cross-validation: techniques, applications, and recent advances. *Advances in Economics, Management and Political Sciences*, v. 99, p. 69–72, 2024.

RAVIKIRAN, H. et al. Optimizing sheep breed classification with bat algorithm-tuned cnn hyperparameters. *SN Computer Science*, Springer, v. 5, n. 2, p. 219, 2024.

ROSENBLATT, F. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, American Psychological Association, v. 65, n. 6, p. 386, 1958.

ROSENFELD, A. Computer vision: basic principles. *Proceedings of the IEEE*, IEEE, v. 76, n. 8, p. 863–868, 1988.

RUSSAKOVSKY, O. et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, Springer, v. 115, p. 211–252, 2015.

SILVA, A. S. et al. Adoção de tecnologias de precisão em fazendas leiteiras no brasil. 2024.

SILVA, L. et al. A new database for breast research with infrared image. *Journal of Medical Imaging and Health Informatics*, American Scientific Publishers, v. 4, n. 1, p. 92–100, 2014.

SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

SINGH, A.; SINGH, P. Image classification: A survey. journal of informatics electrical and electronics engineering. *vol*, v. 1, p. 1–9, 2020.

SZELISKI, R. *Computer vision: algorithms and applications*. [S.l.]: Springer Nature, 2022.

TAN, M.; LE, Q. Efficientnet: Rethinking model scaling for convolutional neural networks. In: PMLR. *International conference on machine learning*. [S.l.], 2019. p. 6105–6114.

TOMAR, V.; BANSAL, M.; SINGH, P. Metaheuristic algorithms for optimization: A brief review. *Engineering Proceedings*, MDPI, v. 59, n. 1, p. 238, 2024.

WANG, R.; NABNEY, I.; GOLBABAEE, M. Efficient hyperparameter importance assessment for cnns. *arXiv preprint arXiv:2410.08920*, 2024.

WOJCIUK, M. et al. Improving classification accuracy of fine-tuned cnn models: Impact of hyperparameter optimization. *Heliyon*, Elsevier, v. 10, n. 5, 2024.

WU, D. et al. Monitoring the respiratory behavior of multiple cows based on computer vision and deep learning. *Journal of Dairy Science*, Elsevier, v. 106, n. 4, p. 2963–2979, 2023.

YANG, X.-S. A new metaheuristic bat-inspired algorithm. In: *Nature inspired cooperative strategies for optimization (NICSO 2010)*. [S.l.]: Springer, 2010. p. 65–74.

YING, X. An overview of overfitting and its solutions. In: IOP PUBLISHING. *Journal of physics: Conference series*. [S.l.], 2019. v. 1168, p. 022022.