UNIVERSIDADE FEDERAL DE SANTA CATARINA
CAMPUS FLORIANÓPOLIS
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE AUTOMAÇÃO E SISTEMAS

Yessica Maria Valencia Lemos

Inspeção automática de defeitos em ovos comerciais usando visão computacional

Florianópolis
2021

Yessica Maria Valencia Lemos

Inspeção automática de defeitos em ovos comerciais usando visão computacional

Dissertação submetida ao Programa de Pós-Graduação em Engenharia de Automação e Sistemas da Universidade Federal de Santa Catarina para a obtenção do título de mestre em Engenharia de Automação e Sistemas.
Orientador: Prof. Marcelo Ricardo Stemmer, Dr.
Coorientador: Prof. Maurício Edgar Stivanello, Dr.

Florianópolis
2021

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Valencia Lemos, Yessica Maria
Inspeção automática de defeitos em ovos comerciais usando visão computacional / Yessica Maria Valencia Lemos ; orientador, Marcelo Ricardo Stemmer, coorientador, Maurício Edgar Stivanello, 2021.
169 p.

Dissertação (mestrado) - Universidade Federal de Santa Catarina, Centro Tecnológico, Programa de Pós-Graduação em Engenharia de Automação e Sistemas, Florianópolis, 2021.

Inclui referências.

1. Engenharia de Automação e Sistemas. 2. Inspeção automática. 3. Defeitos ovos. 4. Visão computacional. I. Stemmer, Marcelo Ricardo. II. Stivanello, Maurício Edgar. III. Universidade Federal de Santa Catarina. Programa de Pós-Graduação em Engenharia de Automação e Sistemas. IV. Título.

Yessica Maria Valencia Lemos

Inspeção automática de defeitos em ovos comerciais usando visão computacional

O presente trabalho em nível de mestrado foi avaliado e aprovado por banca examinadora composta pelos seguintes membros:

Prof. Jomi Fred Hübner, Dr.
Universidade Federal de Santa Catarina

Prof. Eric Aislan Antonelo, Dr.
Universidade Federal de Santa Catarina

Prof. Mário Lúcio Roloff, Dr.
Instituto Federal Catarinense

Certificamos que esta é a **versão original e final** do trabalho de conclusão que foi julgado adequado para obtenção do título de mestre em Engenharia de Automação e Sistemas.

ICP-Edu
Documento assinado digitalmente
Werner Kraus Junior
Data: 24/06/2021 16:21:41-0300
CPF: 531.085.239-53
Verifique as assinaturas em https://v.ufsc.br

Prof. Werner Kraus Junior, Dr.
Coordenador do Programa

ICP-Edu
Documento assinado digitalmente
Marcelo Ricardo Stemmer
Data: 24/06/2021 16:09:33-0300
CPF: 376.093.279-72
Verifique as assinaturas em https://v.ufsc.br

Prof. Marcelo Ricardo Stemmer, Dr.
Orientador

Florianópolis, 2021.

Este trabalho é dedicado a meus pais Bernardo e Maria Margot pelo seu amor e compromisso.

AGRADECIMENTOS

Ao meu pai Bernardo, minha mãe Maria Margot e meu irmão David por todo o amor, compreensão e compromisso que me têm dado. Ao meu namorado Jhon, por nunca desistir e por me acompanhar nessa jornada cheia de altos e baixos. A todos os meus familiares e amigos pelo amor incondicional.

Agradeço ao meu Orientador Prof. Dr. Marcelo Ricardo Stemmer e ao meu coorientador Prof. Dr. Mauricio Stivanello, pela orientação, aconselhamento e apoio contínuo nesta pesquisa. Ao coordenador Prof. Dr. Werner Kraus Junior por todo o apoio e pelo trabalho prestado ao departamento. Ao Sr. Enio Snoeijer pelo comprometimento, gentileza e apoio na gestão administrativa do programa. A todos os docentes do PPGEAS pelo trabalho e compromisso, assim como agradeço a UFSC por me dar a oportunidade de fazer parte desta comunidade acadêmica da qual muito me orgulho. Agradecimentos à CAPES pelo auxílio financeiro e à empresa Plasson pelo apoio e disponibilização dos equipamentos e amostras de ovos utilizados nesta pesquisa.

RESUMO

A produção de ovos em linhas industriais cresceu consideravelmente ao longo dos anos. Essa produção alcança valores superiores a 15.000 ovos por hora, tornando cada vez mais difícil a utilização de operadores humanos para a tarefa de identificação visual e detecção de defeitos. Nesse cenário, as indústrias avícolas, com o objetivo de garantir e aumentar a eficácia e eficiência do processo de controle de qualidade, têm investido em novas soluções tecnológicas. Neste trabalho propõe-se comparar três soluções baseadas em processamento de imagens para realizar a inspeção automática de ovos comerciais, a fim de determinar a solução com o melhor equilíbrio em termos de precisão e velocidade. As abordagens propostas são executadas em um sistema de inspeção visual de ovos com simulação real de linhas de produção avícola. A primeira abordagem consiste na aplicação de técnicas clássicas de processamento de imagens e a segunda e terceira abordagens usam técnicas de aprendizagem profunda: classificação de imagens e segmentação semântica para a classificação do ovo em quatro categorias: normal, sujo, geometricamente anormal e ovos fissurados. Essas abordagens foram validadas com sucesso, alcançando uma acurácia média de 81 %, 85 % e 87 % e um tempo médio de processamento em CPU de 0,04 ms, 0,11 ms e 1,35 ms para as abordagens clássicas, classificação de imagens e segmentação semântica, respectivamente. Considerando os critérios de avaliação definidos para este projeto, obteve-se que as abordagens baseadas em técnicas clássicas apresentam um melhor equilíbrio nas métricas de avaliação, permitindo a detecção de várias características dos ovos como cor, formato e defeitos na superfície da casca.

Palavras-chave: Classificação de ovos, Inspeção Visual Automática, Processamento de Imagens, Visão Computacional.

ABSTRACT

Egg production on industrial lines has grown considerably over the years. This production reaches values above 15.000 eggs per hour, making it increasingly difficult to use human operators for the task of visual identification and defects detection. In this scenario, the poultry industries, in order to guarantee and increase the effectiveness and efficiency of the quality control process, have invested in new technological solutions. In this paper, it is proposed to compare three solutions based on image processing to perform the automatic inspection of commercial eggs in order to determine the solution with a better balance in terms of precision and speed. The proposed approaches are executed in a visual egg inspection system with real simulation of poultry production lines. The first approach consists of applying classical image processing techniques and the second and third approaches make use of deep learning techniques: image classification and semantic segmentation for the classification of the egg into four categories: normal, dirty, geometrically abnormal and cracked eggs. These approaches have been successfully validated achieving an average accuracy of 81 %, 85 % and 87 % and an average CPU processing time of 0,04 ms, 0,11 ms and 1,35 ms for the classical approaches, image classification and semantic segmentation, respectively. Considering the evaluation criteria defined for this project, it was obtained that the approaches based on classical techniques present a better balance in the evaluation metrics, allowing the detection of several features of the eggs such as color, shape, and defects on the shell surface.

Keywords: Egg Classification, Automatic Visual Inspection, Image Processing, Computer Vision.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

O ovo é um dos alimentos mais difundidos e facilmente acessíveis para a população. O Brasil é o sexto maior produtor de ovos do mundo, atrás de outros como China, Estados Unidos, Índia, Japão e México (WEE, 2019). Nos últimos anos, mostrou um crescimento considerável. Estima-se que a produção aumentará 37.5 % até 2028, com uma produção aproximada de 159.461 caixas de 30 dúzias de ovos de acordo com projeções do Departamento de Agronegócio no Brasil (AGRONEGÓCIO, 2020). Este produto representa um alimento rico em nutrientes, fosfolipídios, proteínas, vitaminas, minerais e antioxidantes (BLESSO, 2015). No entanto, a valorização das vantagens nutricionais e funcionais do ovo pelos consumidores depende da qualidade desses produtos oferecidos no mercado. Portanto, é necessário a implementação contínua de programas que garantam altos padrões de qualidade para os ovos, sob a aplicação de boas práticas de produção que certificam o bem-estar dos animais, do meio ambiente e dos trabalhadores.

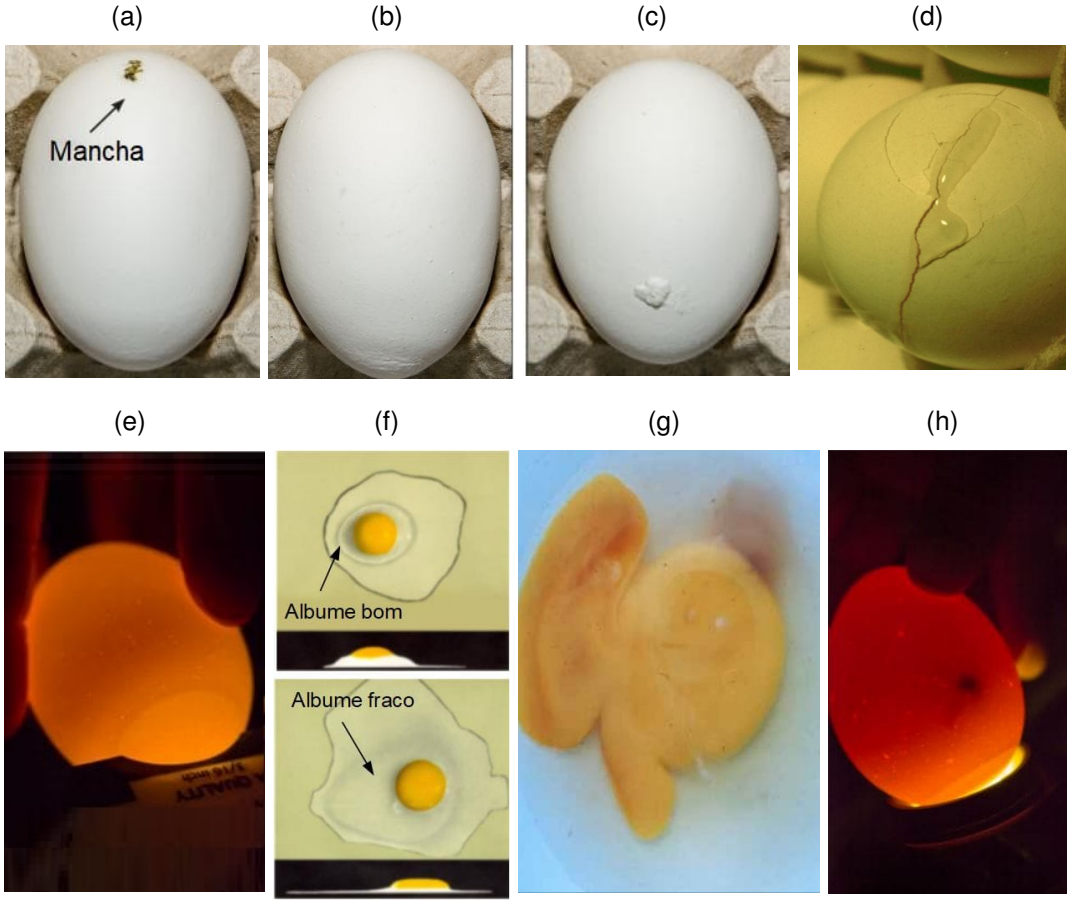
O sistema de processamento de ovos para consumo humano consiste em quatro etapas principais: coleta, lavagem, classificação e embalagem. Especialmente, a etapa de classificação representa um aspecto fundamental da indústria avícola, não apenas por razões econômicas, mas também por razões sanitárias (ARIVAZHAGAN *et al.*, 2013). Nesta etapa, os ovos são submetidos a controle de qualidade por meio de controle físico e de aparência, como tamanho, forma, cor, casca e inspeção dos componentes internos (NARIN *et al.*, 2018). As várias causas de defeitos que podem ocorrer em um ovo estão resumidas no Quadro 1 exemplificadas na Figura 1.

Quadro 1 – Defeitos internos e externos do ovo.

	Fator de qualidade	**Causas de defeitos**
Defeitos externos	Sujeira	- Vestígios de óleos de processamento - Pequenas partículas ou manchas - Sujeira aderente ou material estranho
	Forma	- Forma incomum ou decididamente deformado
	Textura	- Pode ter pequeno depósito de cálcio - Áreas extremamente ásperas que podem apresentar falhas de solidez ou resistência.
	Fissuras	- Casca quebrada ou rachada - Com esvaziamento do conteúdo através do ovo.
Defeitos internos	Célula de ar	- 3/16 polegadas ou menos de profundidade
	Albume branco	- Pode ser fraco e aguado - Pode ser razoavelmente firme
	Gemas	- O contorno não está bem definido - Pode ter grande depósito de cálcio
	Manchas de sangue ou carne	- Pode mostrar manchas finas pronunciadas

Fonte – (ROY *et al.*, 2019).

Figura 1 – Exemplos de defeitos em ovos: (a) Sujeira; (b) Forma; (c) Ovo com depósito de cálcio; (d) Ovo quebrado; (e) Célula de ar; (f) Albume bom e albume fraco; (g) Gema não definida e (h) Mancha de sangue.



Fonte – (PESCATORE; JACOB, 2013) e (CDFA, 2016).

A casca do ovo representa um fator determinante, pois, do ponto de vista do consumidor, serão rejeitados ovos sujos e pálidos com deformidades ou cores desiguais. Além disso, as manchas presentes na casca dos ovos causadas por penas de frango, estrume de frango, podem ser uma fonte de infecção, porque podem conter diferentes tipos de bactérias patogênicas, como *Salmonella, Escherichia colia e Staphylococcus*, que causam doenças graves de origem alimentar (MOR-MUR; YUSTE, 2010).

Alguns defeitos internos e externos são identificados durante o processo de inspeção usando *"Candling"*, uma técnica que consiste em colocar o ovo sobre um buraco e aplicar uma fonte de luz para mostrar os detalhes do ovo através da casca. Essa técnica é amplamente utilizada na indústria avícola para avaliar a qualidade dos ovos, pois fornece um alto contraste entre o primeiro plano e o fundo. No nível industrial, essa técnica quando executada por operadores humanos, não é muito eficaz, pois os trabalhadores enfrentam longas horas de trabalho expostas à luz intensa em ambientes pouco iluminados, como pode ser visto na Figura 2. Isso não apenas produz baixa eficiência de detecção, mas também causa vários problemas de saúde para os

trabalhadores, como fatiga visual, danos visuais progressivos, desconforto físico e mental (SEBASTIÁN *et al.*, 2018).

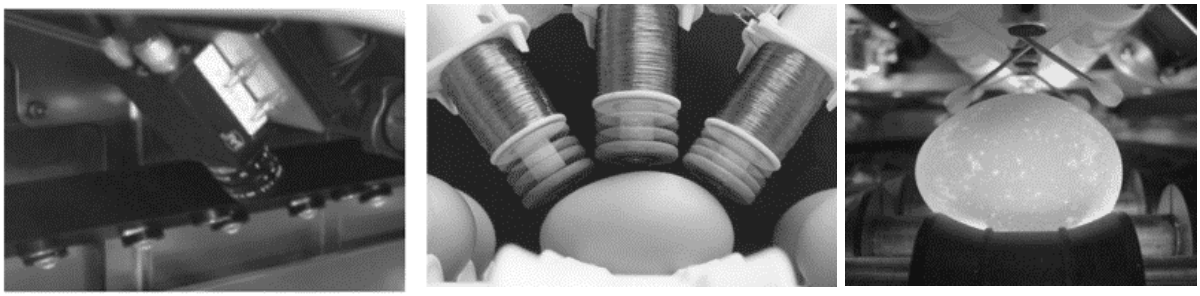
Figura 2 – Identificar ovos com imperfeições usando iluminação inferior.

Fonte – (PLASSON, 2020)

Para automatizar a inspeção visual dos ovos e, assim, reduzir os problemas associados à implementação de técnicas manuais, as empresas investiram em novas tecnologias para detectar defeitos nos ovos usando diferentes abordagens, como técnicas ópticas, mecânicas, elétricas ou acústicas (Figura 3).

Figura 3 – Diferentes tecnologias para detectar defeitos nos ovos usando (a) técnicas ópticas; (b) e (c) mecânicas e acústicas

(a) (b) (c)



Fonte – (MERTENS *et al.*, 2011).

Cada vez mais estão sendo desenvolvidas tecnologias rápidas, confiáveis e totalmente automatizadas que oferecem a possibilidade de avaliar a qualidade de todos os ovos individuais, em vez de colher amostras. A escolha das técnicas depende

da velocidade de inspeção, do custo da instrumentação e da eficiência de classificação necessária (MERTENS *et al.*, 2011). O Quadro 2 mostra uma comparação de três métodos para identificar defeitos em produtos alimentícios. Pode-se observar que o uso de técnicas que empregam visão computacional apresenta um melhor desempenho para esse tipo de prática.

Quadro 2 – Comparação de três métodos de detecção de defeitos em produtos alimentícios.

Método	**Acurácia**	**Velocidade**	**Destrutividade**	**Confiabilidade**
Mecânico	80-90 %	Lenta	Destrutivo	Baixa
Vibracional	Mais de 90 %	Lenta	Destrutivo	Relativamente alta
Visão computacional	Mais de 90 %	Rápida	Não destrutivo	Alta

Fonte – (ABBASPOUR-GILANDEH; AZIZI, 2018).

A principal vantagem das abordagens que utilizam a visão computacional é a sua não destrutividade, isto se deve ao fato de o processo de inspeção ser realizado sem contato direto com o produto, evitando o menor dano possível. Contudo, algumas técnicas de visão computacional são difíceis de implementar, porque muitos dos algoritmos de processamento de imagem são complexos (GUANJUN *et al.*, 2019). Tais algoritmos dependem fortemente da qualidade das imagens, o equipamento requer algumas condições operacionais especiais e, dependendo do algoritmo, o tempo de processamento pode aumentar exponencialmente. Tudo isso dificulta a implementação em ambientes industriais. Por esse motivo, muitas indústrias avícolas continuam implementando o processo de classificação manualmente ou realizando a automação de apenas parte do processo.

Até o momento, diversas abordagens baseadas em análise de imagens têm sido propostas para a detecção e classificação de defeitos em ovos comerciais. De acordo com as informações coletadas no mapeamento sistemático da literatura realizado neste trabalho, verificou-se que a maioria dessas abordagens utiliza métodos clássicos de processamento de imagens, que conseguem projetar aplicativos de sucesso com baixo custo computacional, porém, podem se tornar complexos, pois requerem várias etapas diferentes, como remover o fundo, manter a região de interesse, aplicar filtros, entre outros. Portanto, há uma demanda por novas técnicas que possam lidar com essas complexidades. Nos últimos anos, o aprendizado profundo foi desenvolvido para tarefas automatizadas baseadas em visão, como reconhecimento de padrões e classificação de imagens (LECUN *et al.*, 2015). Redes Neurais Convolucionais (*CNNs*) são um dos métodos de aprendizado profundo mais importantes e bem-sucedidos no campo da análise de imagens, em que várias camadas são combinadas e treinadas de forma eficiente. CNNs provaram ser ferramentas promissoras no campo da indústria agrícola e de alimentos (SLADOJEVIC *et al.*, 2016), (SANTOS FERREIRA *et al.*, 2017), (FAROOQ; SAZONOV, 2017), (SHIMIZU *et al.*, 2017). No entanto, algumas

dessas abordagens podem ser caras do ponto de vista computacional. Essas incertezas nas abordagens levantam alguns questionamentos quanto à sua implementação em ambientes industriais, especialmente no processo de classificação e detecção de defeitos, questões como: Entre as abordagens clássicas e baseadas em aprendizagem profunda, qual abordagem tem o melhor equilíbrio entre precisão e velocidade?; Como as abordagens de aprendizado profundo podem resolver a complexidade das abordagens clássicas?; As abordagens baseadas em aprendizagem profunda podem ser implementadas em sistemas de inspeção de ovos em tempo real? Com o objetivo de responder a essas questões, esta pesquisa apresenta uma comparação entre abordagens clássicas e abordagens baseadas em aprendizagem profunda, aplicadas a um problema de classificação de ovos em quatro categorias, ovos normais, ovos sujos, ovos geometricamente anormais e ovos fissurados, destinado a uso industrial.

1.1 OBJETIVOS

Nas seções abaixo estão descritos o objetivo geral e os objetivos específicos para esta pesquisa.

1.1.1 Objetivo Geral

O objetivo desta pesquisa é comparar o uso de técnicas clássicas de processamento e análise de imagens com o uso de técnicas de aprendizagem profunda em um sistema de classificação de ovos comerciais em quatro categorias: ovos normais, ovos sujos ou manchados, ovos geometricamente anormais e ovos fissurados, com furos ou grandes danos na casca.

1.1.2 Objetivos Específicos

a. Projetar um protótipo para inspeção visual de ovos comerciais

b. Implementar um método baseado em técnicas clássicas de processamento e análise de imagens para a classificação de defeitos em ovos comerciais

c. Implementar um método baseado em técnicas de aprendizagem profunda usando duas abordagens: classificação de imagem e segmentação semântica para a classificação de defeitos em ovos comerciais

d. Comparar o desempenho dos métodos propostos

1.1.3 Estrutura do documento

Este trabalho está organizado da seguinte forma: O capítulo 2 apresenta a fundamentação teórica necessária para o desenvolvimento do trabalho. O Capítulo 3

apresenta os trabalhos relacionados à detecção e classificação de defeitos em ovos comerciais usando visão computacional. Posteriormente, no Capítulo 4, é descrito como o trabalho foi desenvolvido, os materiais e as ferramentas utilizadas. Os resultados são resumidos e analisados no Capítulo 5. Finalmente, no Capítulo 6, são apresentadas as conclusões do trabalho, bem como sugestões para pesquisas futuras.

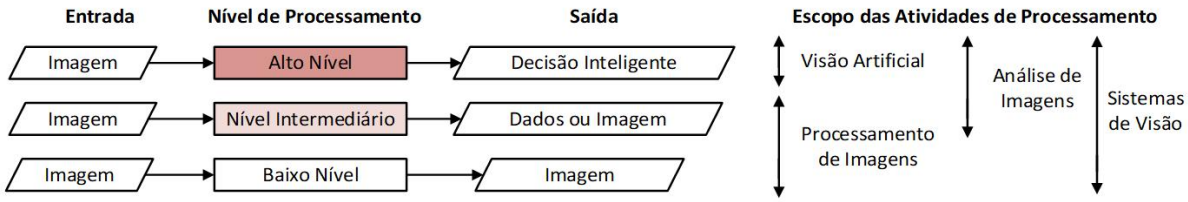
2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, são discutidos os principais conceitos teóricos que servirão de base para o desenvolvimento do sistema de inspeção visual de ovos comerciais.

2.1 SISTEMAS DE VISÃO COMPUTACIONAL

Os sistemas de visão computacional compreendem vários níveis de processamento como mostrado na Figura 4. O processamento de imagens e a visão artificial desempenham um papel fundamental no desenvolvimento desses sistemas. A visão artificial, parte da inteligência artificial, é um conjunto de teorias, técnicas e métodos que permitem simular o processo de visão biológica dos seres humanos e a capacidade de extrair e analisar automaticamente as informações das imagens.

Figura 4 – Níveis de processamento dos sistemas de visão computacional.



Fonte – (STIVANELLO *et al.*, 2019).

Entre os elementos mais importantes dos sistemas de visão estão: o sistema de iluminação, sistema de aquisição e o sistema de processamento. Algumas das principais características a serem considerados nesses sistemas são brevemente descritas a seguir.

2.1.1 Sistema de iluminação

A iluminação pode ser considerada como a parte mais crítica de um sistema de visão. Nestes sistemas, três fatores principais devem ser considerados: intensidade, direção e fonte de iluminação. A intensidade é um fator que pode gerar grandes variações nas imagens. Por exemplo, uma forte intensidade permite obter grandes contrastes nas áreas iluminadas e nas áreas sombreadas. No entanto, por possuir um alcance limitado, alguns detalhes de as imagens são perdidos. Por outro lado, quando a intensidade é baixa, permite apreciar melhor os detalhes nas áreas iluminadas e sombreadas, mas perde os detalhes associados às texturas. Em relação à direção da luz, em geral, existem três configurações principais:

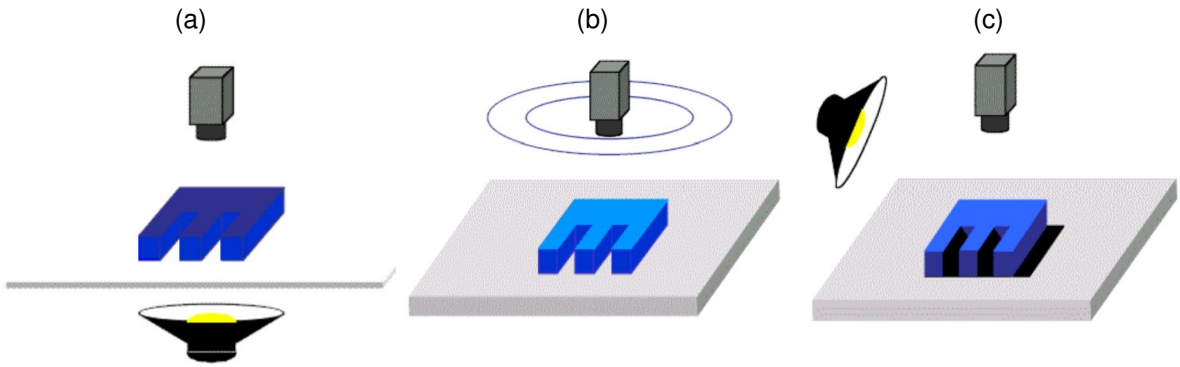
- Iluminação traseira: também é conhecida como luz de fundo difusa. Nessa configuração, o objeto é colocado entre a câmera e a luz, conforme mostrado na

Figura 5a. Esse tipo de iluminação permite um alto contraste entre os objetos e o fundo.

- Iluminação frontal: o objeto é iluminado pela frente, ou seja, a luz cai diretamente sobre o objeto (Figura 5b). Permite visualizar as características externas de objetos como forma, cor ou superfície. É o tipo de iluminação mais utilizado, mas às vezes não é obtido um bom contraste entre o objeto e o fundo, devido ao aparecimento de sombras e reflexos.

- Iluminação direcional: nesta configuração, a luz é orientada em direção ao objeto a partir de qualquer posição no espaço (Figura 5c). Esse tipo de iluminação permite obter informações tridimensionais (3D) sobre os objetos, pois gera facilmente sombras que aumentam o contraste nas partes com variação de profundidade.

Figura 5 – Tipos de iluminação: a) iluminação traseira, b) iluminação frontal e c) iluminação direcional.



Fonte – (GONZALEZ; WOODS, 2004).

O último fator, está associado à fonte de iluminação, entre as mais comuns estão:

- Lâmpada incandescente: foi a primeira fonte de luz da energia elétrica e é a fonte mais comum de iluminação. É um dispositivo que produz luz aquecendo um filamento metálico pelo efeito Joule até ficar vermelho branco, graças à passagem de eletricidade. Sua principal vantagem é que ela possui uma variedade de potências, e algumas desvantagens são a geração de calor excessivo, um tempo de vida relativamente curto e a redução do brilho ao longo do tempo.

- Fluorescente: fornece uma luz brilhante sem sombras. Esse tipo de iluminação é mais eficaz do que a iluminação por lâmpadas incandescentes e fornece uma luz mais difusa; portanto, recomenda-se o uso em objetos altamente refletivos. Geralmente, as lâmpadas fluorescentes padrão não são usadas devido ao seu efeito

tremeluzente, portanto, lâmpadas fluorescentes que operam em alta frequência são frequentemente usadas.

- *Light-Emitting Diode (LED)*: é uma fonte de estado sólido que emite luz quando a eletricidade é aplicada a um semicondutor. Este tipo de iluminação fornece luz difusa muito útil de intensidade moderada. Possui muitas vantagens sobre a iluminação tradicional, como vida útil longa, baixo consumo de energia, baixa geração de calor, tamanho pequeno, econômico, resposta rápida, robustez e menor sensibilidade às vibrações.
- Laser: é geralmente usado para medição de profundidade, detecção de irregularidades na superfície ou reconhecimento 3D. Sua principal desvantagem é que não é eficiente em superfícies de absorção de luz.

2.1.2 Sistema de aquisição

As câmeras são os principais componentes dos sistemas de aquisição. Uma câmera é composta de uma lente, um elemento fotossensível, um circuito de processamento e uma fonte de alimentação. A função básica da lente é realizar a transformação do feixe de luz, permitindo que a imagem do alvo seja projetada na superfície fotossensível do sensor de imagem. A lente também permite definir o campo de visão da câmera a partir da distância focal. Essa distância focal pode ser fixa ou variável, também conhecida como *varifocal*. Intuitivamente, a principal diferença é que as lentes *varifocais* permitem definir a distância focal desejada. O elemento fotossensível é um sensor de imagem, chip óptico e chip fotossensível. Este sensor é uma peça geométrica de cristal de silício com dezenas de milhares de pontos fotossensíveis que podem detectar e registrar informações sobre a luz externa e depois convertê-las em informações digitais. O elemento fotossensível pode ser classificado em dois tipos: um *Charge Coupling Device (CCD)* e um *Complementary Metal Oxide Semiconductor (CMOS)*. O circuito de processamento converte a informação digital obtida pelo elemento fotossensível em um arquivo de dados de imagem digital (BI, s.d.).

2.1.3 Sistema de processamento

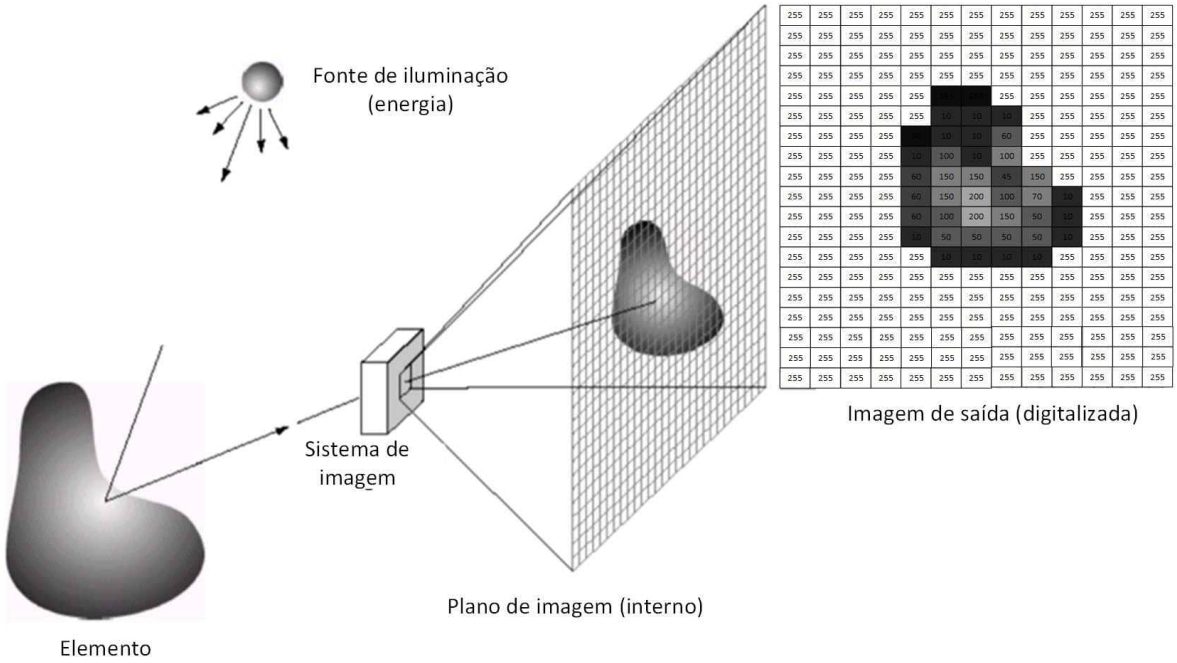
O sistema de processamento é composto por hardware que pode ser de diferentes arquiteturas, este se encarrega de executar softwares especializados com diferentes algoritmos que compõem um fluxo de processamento de imagens.

2.1.4 Aquisição e representação de imagens digitais

Nesse estágio, é obtida uma imagem digitalizada do objeto, representada matematicamente como uma matriz bidimensional de valores inteiros, em que cada valor é

conhecido como pixel. Nas imagens em tons de cinza, o valor de cada pixel é representado por meio de um escalar, enquanto nas imagens coloridas o valor de cada pixel é um vetor de três coordenadas, cada uma das quais especifica o grau de influência das cores. Normalmente, são usadas imagens de 8 bits escala [0,255], porque o olho humano não é sensível o suficiente para diferenças de mais de 256 níveis de intensidade para uma cor. Portanto, para imagens em tons de cinza, o intervalo para os valores de pixel é [0-255] como mostrado na Figura 6, enquanto para imagens coloridas, os valores dos elementos da matriz assumem valores de ([0-255], [0-255], [0-255]).

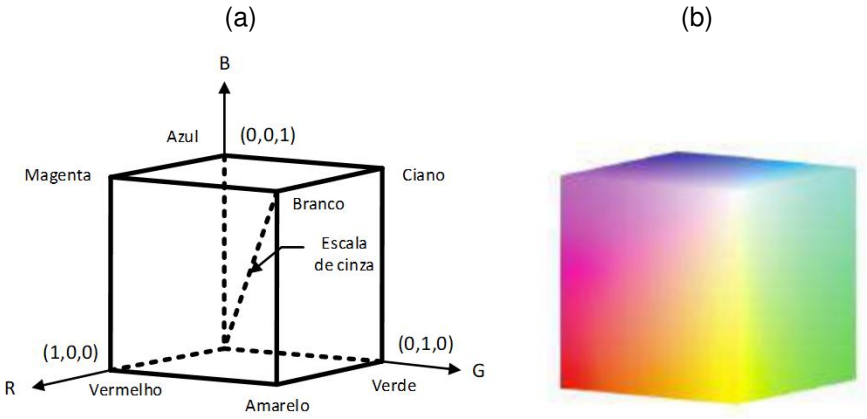
Figura 6 – Processo de aquisição de imagem digital.



Fonte – Adaptado de (GONZALEZ; WOODS, 2004).

Em uma imagem colorida, as cores são representadas de forma padronizada a partir de modelos de cores, um dos modelos mais comuns usados no processamento de imagens é o modelo *Red-Green-Blue (RGB)*. O modelo *RGB* é um modelo de cores aditivo que usa uma mistura de 3 componentes espectrais primários vermelho R, verde G e azul B. Este modelo é baseado em um sistema de coordenadas cartesianas que forma um cubo (Figura 7) em que os valores R, G e B estão em 3 cantos; os valores ciano, magenta e amarelo estão nos outros 3 cantos; preto está na origem e branco está no vértice oposto à origem. A escala de cinza é uma linha que liga os pontos preto e branco. As diferentes cores são pontos no cubo ou dentro dele e são definidas por meio de vetores que se estendem desde a origem.

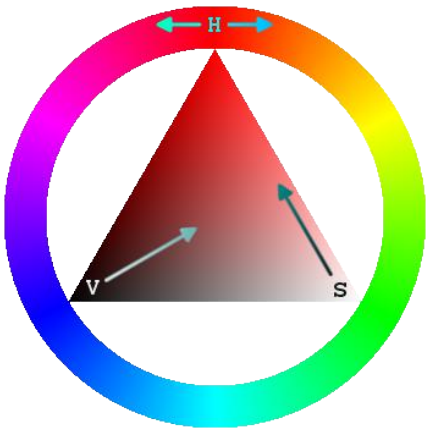
Figura 7 – (a) Esquema do cubo de unidades de cores *RGB* normalizado mostrando cores primárias e secundárias nos vértices. (b) O cubo de cores *RGB*.



Fonte – (GONZALEZ; WOODS, 2004).

Outro modelo muito útil no processamento de imagem é o modelo *Hue-Saturation-Value (HSV)*, pois permite não só obter informações de cor, mas também de saturação e brilho das imagens. No modelo *HSV*, *H* representa o matiz, *S* a saturação e *V* o valor. Também chamado *HSB*, onde *B* é o brilho. O matiz ou tom é a principal qualidade de uma cor. Os tons são todas as cores da roda de cores (Figura 8), primárias, secundárias e intermediárias, sem misturar com preto ou branco. Deslocar para a esquerda ou direita na roda de cores produz uma alteração no tom. A saturação refere-se à quantidade de luz branca misturada com um tom, e o brilho refere-se à capacidade de uma cor refletir luz branca.

Figura 8 – Roda de cores associada a tons.

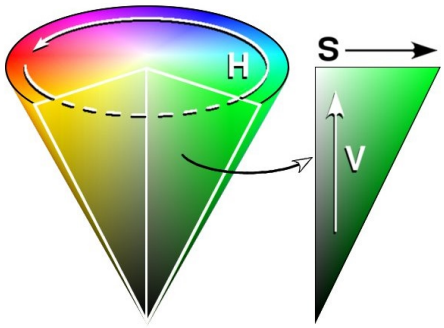


Fonte – (LUIS CABERA, 2011).

O modelo *HSV* usa um sistema de coordenadas cilíndricas, em que o espaço de cores é definido de acordo com uma pirâmide de base hexagonal (Figura 9). Normalmente, o eixo horizontal do triângulo indica saturação, enquanto o eixo vertical

corresponde ao valor da cor. Dessa maneira, uma cor pode ser escolhida pegando primeiro o tom de uma região circular e depois selecionando a saturação e o valor de cor desejado da região triangular.

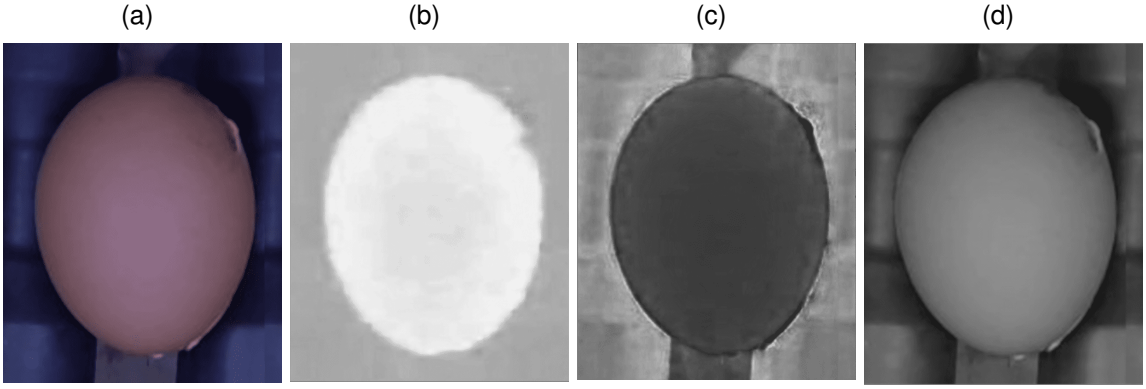
Figura 9 – Representação gráfica do modelo *HSV*.



Fonte – (WIKIPEDIA, 2020).

0 componente *H* é representado como um grau de ângulo cujos valores possíveis variam de 0° a 360° também podem ser normalizados de [0 a 100] %. Cada valor corresponde a uma cor. Exemplos: 0° é vermelho, 60° é amarelo e 120° é verde. O componente *S* é representado como a distância do eixo do brilho preto-branco. Os valores possíveis variam de [0 a 100] %, neste caso, quanto menor a saturação de uma cor, mais acinzentada e descolorida será. Finalmente, o componente *V* representa a altura no eixo preto-branco. Os valores possíveis variam de [0 a 100] %. Onde 0 é sempre preto e dependendo da saturação, 100 pode ser branco ou uma cor mais ou menos saturada. Um exemplo dos componentes *H-S-V* de uma imagem em formato *RGB* é mostrado na Figura 10.

Figura 10 – (a) Imagem em formato *RGB*, (b) componente *H*, (c) componente *S* e (d) componente *V*



Fonte – O autor.

2.2 ALGORITMOS CLÁSSICOS DE PROCESSAMENTO E ANÁLISE DE IMAGENS

Os algoritmos clássicos de processamento de imagem são específicos para cada tarefa particular. David Marr em (GEISLER, 1983) apresentou as bases gerais para a classificação e sistematização dos métodos de reconhecimento de padrões e análise de imagens. Ele propôs um processo sequencial de quatro etapas para extrair as informações relevantes das imagens e representá-las simbolicamente. Essas etapas são (RATEKE, 2019):

- *Filtragem e Pré-processamento:* não são esperadas modificações profundas nas imagens, apenas são atenuadas ou aprimoradas.
- *Condicionamento:* espera-se que uma nova imagem simplificada seja gerada, onde as características mais importantes das imagens são destacadas.
- *Rotulação:* início da etapa de interpretação. Nesta etapa, a imagem é convertida em um modelo que descreve a imagem a partir de um conjunto de dados simbólicos.
- *Interpretação:* classificação e interpretação dos dados, as descrições das estruturas das imagens são traduzidas, apresentadas em linguagens ou esquemas de representação com semântica no contexto da aplicação.

Além disso, uma imagem pode ser visualizada em três domínios diferentes nos quais seus dados podem ser interpretados:

- Valor: cada pixel é considerado independentemente.
- Espaço da imagem: cada pixel apresenta um contexto: nesse caso, os pixels são analisados juntos para determinar diferenças ou semelhanças entre eles.
- Frequência: são considerados o valor do pixel, o contexto do pixel no espaço e a variação do pixel no espaço.

Combinando esses dois esquemas de classificação, é obtido o Quadro 3, que classifica e contextualiza as várias famílias de algoritmos de processamento de imagem. Segue-se uma descrição geral de alguns dos algoritmos utilizados nas técnicas clássicas de análise de imagem. Grande parte dessa conceitualização foi adquirida de (VON WANGENHEIM, 2019a, 2019c, 2019d).

Quadro 3 – Classificação de algoritmos para processamento de imagem.

	Filtragem e Pré-processamento	**Condicionamento**	**Rotulação**	**Interpretação**
Tipo	Imagem em imagem	Imagem em imagem	Imagem em modelo	Modelo em modelo
Valor	- Janelamento - Transformações de escala	- Limiarização - Operações matemáticas		

Espaço	- Eliminação de chuvisco
	- Morfologia matemática
	  - Erosão
	  - Abertura
	  - Fechamento
	  - Etc.
		- Detecção de bordas
		  - Filtros de convolução
		- Segmentação
		- Morfologia matemática
		  - Esqueletonização
			- Todos tipos de classificadores espaciais
			- Métodos de classificação estatísticos
			- Classificadores baseados em regras
			- Redes neurais
			- *Consistent labeling*
				- Estruturas de dados espaciais
				  - *Octree*
				  - *Quadtree*
				  - Modelos de facetas
				  - *Boundary models*
				- Gramáticas
				- Grafos
				- Redes semânticas
				- Redes neurais

Frequência	- Fourier
	  - Passa-alta
	  - Passa-baixa
	  - Passa-faixa
	- Wavelets
		- Fourier
		  - Eliminação de faixas
		  - Eliminação de frequências
		  - Agrupamento por faixas de frequência
		- Wavelets (IDER)
			- Fourier
			  - Classificação de canais
			- Wavelets
			  - Classificação de grupos de coeficientes	

Fonte – Adaptado de (RATEKE, 2019).

2.2.1 Métodos no domínio de valor

No domínio do valor, existem vários grupos de operações, um dos quais é o grupo de operações matemáticas, onde é possível adicionar, subtrair, multiplicar e dividir operadores escalares com cada pixel da imagem, permitindo aumentar ou diminuir a sua intensidade. Também é possível aplicar essas operações entre imagens, os valores resultantes são truncados para os valores máximos ou mínimos da representação, como [255-0], respectivamente, para uma imagem de 8 bits. Outro grupo de operações é o grupo de operações lógicas (*OR, AND, XOR, NOT*) que também podem ser aplicadas a partir de um operador escalar ou de uma imagem para outra.

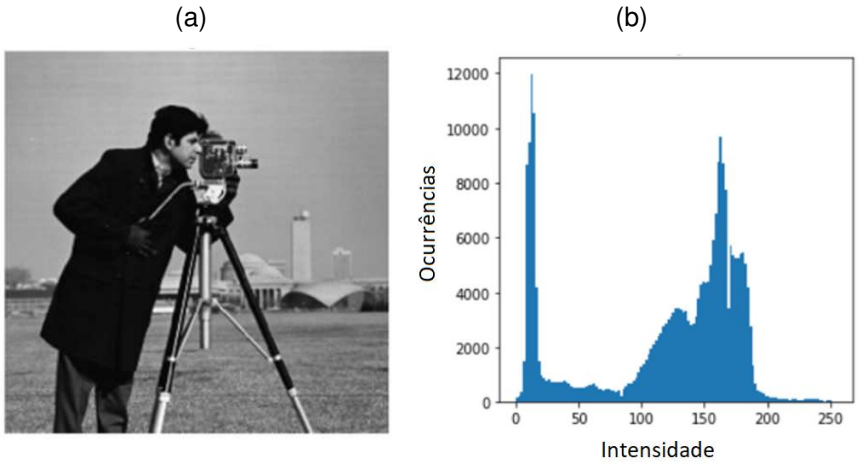
Uma das operações mais usadas no processamento de imagens são as operações de limiarização, basicamente consiste em comparar cada pixel com um valor limiar e atribuir uma determinada condição para cada caso. Essa operação é amplamente usada para binarizar as imagens, por exemplo, na operação de *Limiar acima*, pixels cujo valor for $\geq$ ao limiar recebem a cor branco e o restante a cor preto, um exemplo de binarização com limiar de 100 é mostrado na Figura 11.

Outra das operações que é necessário mencionar no domínio do valor são as operações estatísticas, nas quais se destacam os histogramas. Um histograma determina o número de pixels com o mesmo valor (0 a 255) presente na imagem; é representado na forma de um vetor de 256 posições que contém esses valores. A Figura 12 mostra um exemplo de um histograma. Também é possível usar operações combinadas, como no caso da *limiarização com histograma,* na qual o limiar é definido de acordo com os pontos nodais do histograma, um exemplo disso é mostrado na Figura 13.

Figura 11 – Operação *Limiar acima*.

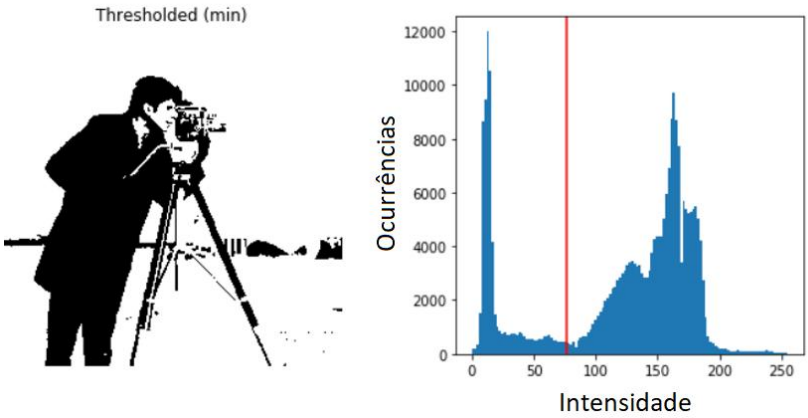
Fonte – Imagem original extraída do banco de dados *scikit-image* (SCIKIT-IMAGE, 2020).

Figura 12 – (a) Imagem original e (b) histograma da imagem.



Fonte – Imagem original extraída do banco de dados *scikit-image* (SCIKIT-IMAGE, 2020).

Figura 13 – Limiares com histograma bimodal com base no mínimo global.



Fonte – Imagem original extraída do banco de dados *scikit-image* (SCIKIT-IMAGE, 2020).

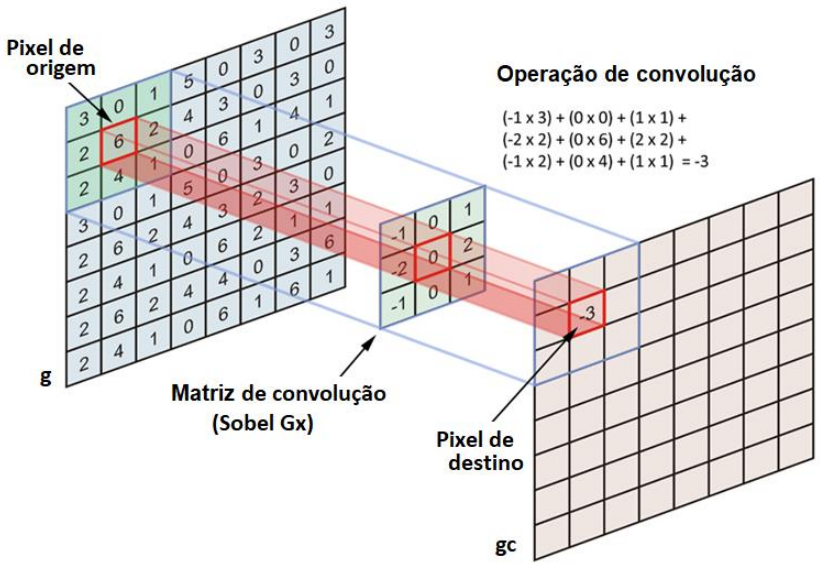
2.2.2 Métodos no Domínio do Espaço: principais conceitos

Cinco conceitos serão analisados: convolução, morfologia matemática, operações de detecção de bordas, segmentação e rotulação .

2.2.2.1 Convolução

A convolução é uma operação entre duas matrizes, uma das quais é a imagem e a outra é uma matriz chamada matriz de convolução, elemento estruturante ou *kernel*. A matriz de convolução representa qualquer função matemática e é aplicada a cada pixel *g(x, y)* da imagem e sua vizinhança imediata, resultando em uma nova imagem *gc(x, y)*. O resultado de uma convolução é obtido da soma das multiplicações de cada elemento da matriz pela região da imagem abaixo dela, esse resultado substitui o valor do pixel no qual a matriz foi aplicada (consulte a Figura 14).

Figura 14 – Exemplo de convolução de uma imagem com uma matriz de detecção de borda (Sobel).



Fonte – Adaptado de (VON WANGENHEIM, 2019b).

2.2.2.2 Morfologia Matemática

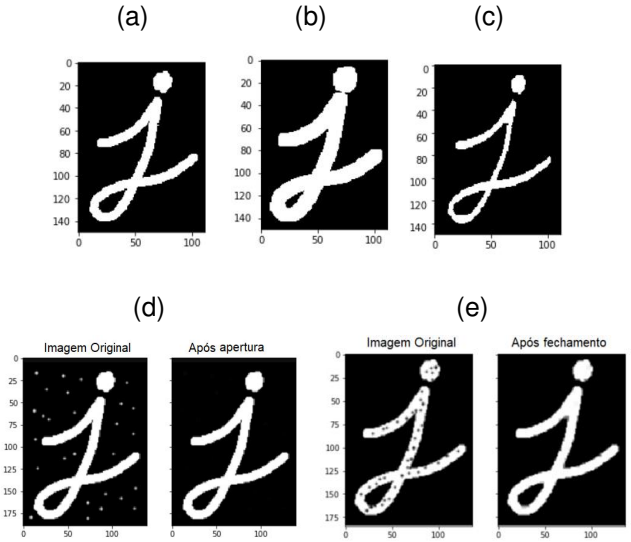
Operações morfológicas são métodos para processar imagens binárias com base em formas. Essas operações recebem uma imagem binária como entrada e resultam em uma imagem binária como saída. Entre as operações morfológicas mais utilizadas estão:

- Dilatação binária: a dilatação expande uma imagem utilizando um elemento estruturante. Consiste em passar o elemento estruturante através de todos os pixels da imagem e, se o valor do pixel da imagem sob o elemento central for diferente

de zero, são copiados todos os valores não-zero do elemento estruturante para a imagem resultado (Figura 15b).

- Erosão binária: a erosão encolhe uma imagem de acordo com critérios dados por um elemento estruturante. Consiste em passar o elemento estruturante por todos os pixels da imagem, se nenhum valor dos pixels da imagem sob os valores não-nulos do elemento estruturante for zero, se põe um valor 1 (ou 255) na posição R da imagem resultado (Figura 15c).
- Abertura (*opening*): suaviza o contorno de uma imagem, quebra estreitos e elimina proeminências delgadas, é usada também para remover ruídos da imagem e abrir pequenos vazios ou espaços entre objetos próximos numa imagem. É obtida aplicando uma erosão seguida de uma dilatação com o mesmo elemento estruturante (Figura 15d).
- Fechamento (*closing*): funde pequenas quebras e alarga golfos estreitos, elimina pequenos orifícios, preenche ou fecha os vazios. É obtida aplicando uma dilatação seguida de uma erosão com o mesmo elemento estruturante (Figura 15e).

Figura 15 – (a) Imagem original, (b) erosão da imagem (a), (c) dilatação da imagem (a), (d) abertura e (e) fechamento.



Fonte – O autor.

2.2.2.3 Operações de Detecção de Bordas

Uma borda é definida como uma variação no nível de cinza, quando a magnitude da imagem muda abruptamente. Um operador sensível a essas alterações, assim como o operador derivado, funciona como um detector de borda. Basicamente, uma derivada é a taxa de variação de uma função, a taxa de variação dos níveis de cinza em uma

imagem é maior próximo às bordas e menor em áreas constantes. Obtendo os valores da intensidade da imagem e os pontos em que a derivada é um ponto máximo, é possível determinar as bordas.

Para operadores de detecção de borda, são utilizados operadores de convolução que usam máscaras de convolução projetadas de forma a fornecer resposta máxima quando superam uma variação abrupta na intensidade do sinal. Um desses operadores é o Laplaciano, este é um operador de derivação isotrópica, ou seja, é independente da direção da descontinuidade na imagem. O Laplaciano *L(x, y)* de uma imagem com valores de intensidade de pixel *I(x, y)* é dado por:

$$L(x, y) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2} \tag{1}$$

Como a imagem de entrada é representada como um conjunto de pixels discretos, um *kernel* de convolução discreta deve ser encontrado que se aproxime das segundas derivadas na definição do Laplaciano. Dois exemplos de filtros Laplacianos comumente usados são mostrados na Figura 16.

Figura 16 – Duas aproximações discretas comumente usadas para o filtro Laplaciano.

0	−1	0
−1	4	−1
0	−1	0

−1	−1	−1
−1	8	−1
−1	−1	−1

Fonte – (R. *et al.*, 2003).

O Laplaciano geralmente não é usado diretamente na prática, pois é muito sensível ao ruído. Uma maneira de contornar isso é usar um suavizado gaussiano antes de aplicar o filtro laplaciano conhecido como *Laplaciano do Gaussiano (LoG*) que reduz os componentes de ruído de alta frequência antes da etapa de diferenciação. A função2-D *LoG* centrada em zero e com desvio padrão gaussiano σ tem a forma:

$$LoG(x, y) = -\frac{1}{\pi\sigma^4}\left[1 - \frac{x^2+y^2}{2\sigma^2}\right] e^{\frac{x^2+y^2}{2\sigma^2}} \tag{2}$$

Conforme o Gaussiano fica cada vez mais estreito, o *kernel LoG* se torna o mesmo que os *kernels* Laplacianos simples. Isso ocorre porque a suavização com um Gaussiano muito estreito (σ <0.5 pixels) em uma grade discreta não tem efeito, é por isso que um valor adequado de σ deve ser determinado de modo a não perder muitos detalhes das bordas ou trazer muito ruído.

2.2.2.4 Segmentação

A segmentação visa dividir a imagem em suas diversas partes constituintes ou segmentos. Os algoritmos de segmentação baseiam-se principalmente em duas

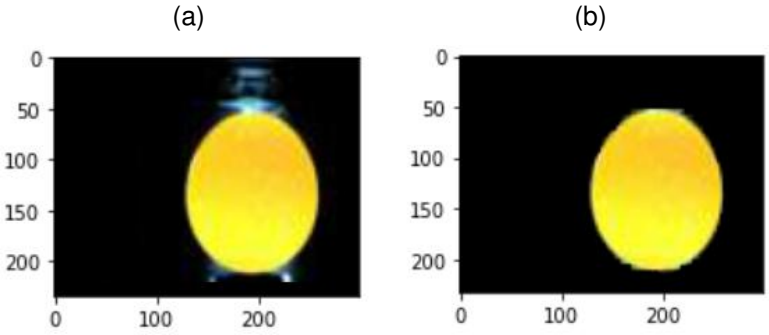
propriedades do nível de intensidade luminosa das imagens:

a. Por descontinuidade: divisão da imagem de acordo com as mudanças abruptas do nível de intensidade luminosa de seus pontos. Permite o realce de cantos e bordas de objetos e regiões. É utilizada quando se buscam por linhas, pontos ou outras formas caracterizadas pelas bordas.

b. Por similaridade: divisão da imagem de acordo com padrões de similaridade encontrados em suas regiões seja por nível de intensidade luminosa ou textura. Utilizadas quando se busca pelos pontos internos dos objetos (STIVANELLO *et al.*, 2019).

Um exemplo de segmentação por similaridade é mostrado na Figura 17, na qual a imagem resultante (Figura 17b) mostra apenas os níveis de intensidade luminosa que estão em um intervalo fixo de valores, neste caso entre [80, 0, 20] e [100, 255, 255] em formato *HSV*.

Figura 17 – (a) Imagem original, (b) imagem segmentada.



Fonte – Imagem original extraída de (DEHROUYEH *et al.*, 2010).

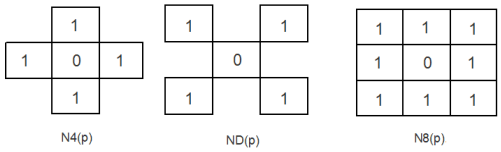
2.2.2.5 Rotulação

Os métodos de rotulação extraem partes de uma imagem e associam rótulos a estas partes pixel-a-pixel. São úteis para dividir imagens em pedaços, o que permite identificar as diferentes regiões da imagem. Um dos métodos mais simples de rotulação é o *método de cálculo de componentes conectados*, utilizado para extrair os diferentes objetos de uma imagem de acordo com alguma característica desses objetos, normalmente a sua cor.

Um componente conectado é um conjunto finito de pixels conectados que compartilham uma propriedade V específica. Pode-se dizer que dois pixels (p e q) estão conectados se houver um caminho de p a q e, se todos os pixels no caminho tiverem a propriedade V, isso permitirá que os pixels relacionados tenham o mesmo rótulo e os componentes da imagem sejam representados com rótulos diferentes. Para definir um componente conectado, os seguintes fatores devem ser considerados:

- Vizinhança: os pixels devem estar relacionados seja N4, ND ou N8 (Figura 18).
- Adjacência: além de ser um vizinho, o pixel deve possuir um valor de intensidade luminosa “semelhante”.
- Conectividade: é gerada quando há um caminho de pontos vizinhos e adjacentes.
- Região: é gerada quando todos os pontos de uma área da imagem são conexos.

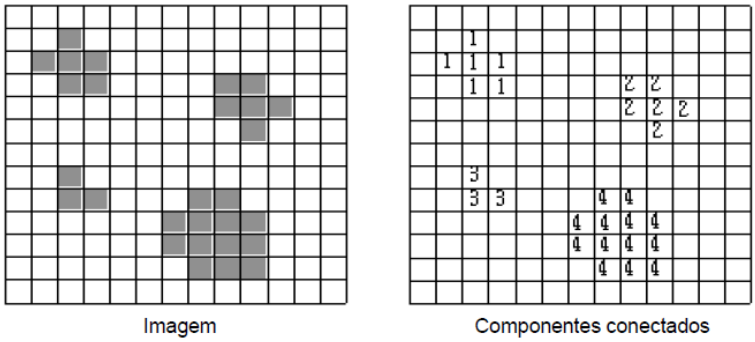
Figura 18 – Vizinhança e adjacência.



Fonte – (STIVANELLO *et al.*, 2019).

Este método localiza todos os componentes conectados de uma imagem e atribui um identificador exclusivo a todos os pontos que compõem cada componente. Os pontos de um componente conectado são normalmente interpretados como objetos ou regiões de um objeto. Um exemplo é mostrado na Figura 19.

Figura 19 – Método de cálculo de componentes conectados.

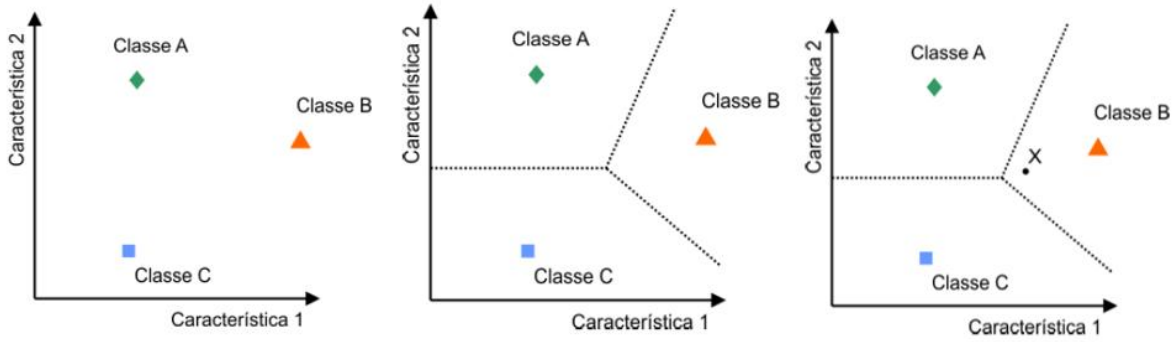


Fonte – (STIVANELLO *et al.*, 2019).

2.2.3 Métodos no Domínio do Espaço, classificadores

Esta seção descreve algumas convenções gerais dos classificadores. Basicamente, a classificação consiste em atribuir uma classe a cada nova amostra; para isso, cada classe deve ter um conjunto de descritores que a caracterizem, a fim de criar um espaço multidimensional que possa determinar possíveis fronteiras entre elas. Um exemplo é mostrado na Figura 20, onde existem três classes (A, B, C) e é necessário estabelecer a qual classe *x* pertence.

Figura 20 – Espaço multidimensional de descritores que representam cada classe.



Fonte – (STIVANELLO *et al.*, 2019).

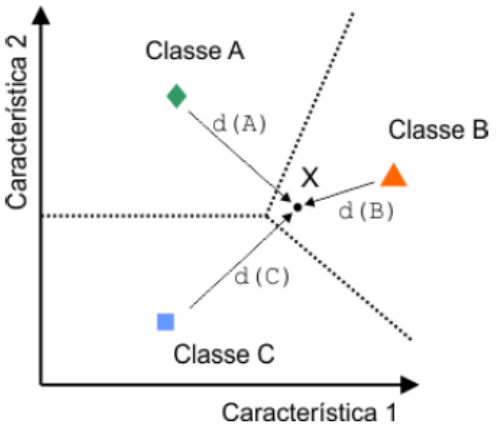
Um dos classificadores mais simples que podem ser usados para resolver esse tipo de problema, são os *classificadores por distância mínima*, nos quais cada imagem/amostra é classificada com base na distância entre seus descritores e os descritores da classe correspondente no espaço multidimensional de descritores. Cada classe é representada por um vetor médio:

$$m_j = \frac{1}{N_j} \sum_{x \in w_j} x \ para \ j = 1, 2, ..., M \tag{3}$$

Onde N_j é o número de vetores/amostras de treinamento da classe W_j.

Pode-se então atribuir uma amostra X a uma dada classe calculando sua proximidade para a cada W_j e selecionando a classe de menor distância. Desta forma, a distância é definida como sendo um índice de similaridade. Normalmente é utilizada a distância euclidiana. No exemplo mostrado na Figura 21, a distância entre a classe B e x é a menor. Desta forma, x é classificado como pertencente a esta classe.

Figura 21 – Exemplo de uso do classificador por distância mínima.



Fonte – (STIVANELLO *et al.*, 2019).

Outros classificadores um pouco mais complexos usam técnicas de aprendizado de máquina. Essas técnicas consistem em automatizar, através de diferentes algoritmos, a identificação de padrões ou tendências nos dados. Para isso, é necessário treinar um modelo que possa aprender como os dados se comportam. O aprendizado nos classificadores pode ser feito de duas maneiras muito diferentes:

- Aprendizado supervisionado: o aprendizado supervisionado requer um conjunto de padrões pelos quais sua verdadeira classe é conhecida. Este conjunto é chamado de conjunto de treinamento que foi rotulado anteriormente. Essa tarefa geralmente é executada por um especialista na área em que será executado o reconhecimento.

- Aprendizado não supervisionado: o aprendizado não supervisionado é realizado a partir de um conjunto de padrões dos quais sua classe não é conhecida. Basicamente, significa encontrar agrupamentos. O objetivo geralmente é verificar a validade do conjunto de classes para uma classificação supervisionada. As técnicas utilizadas são frequentemente chamadas de métodos de agrupamento ou *clustering*.

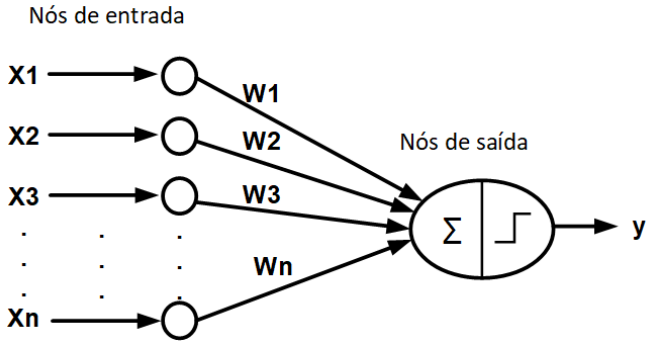
A *Rede Neural Artificial (RNA)* é um exemplo de um classificador amplamente conhecido, que apresentam uma estrutura composta por processadores simples chamados nós ou neurônios, conectados por meio de canais ou conexões de comunicação. Cada neurônio possui uma quantidade de memória local, operando apenas com seus dados locais e nas entradas que recebe através dessas conexões (SOBRADO MALPARTIDA, 2003).

Em geral, as redes neurais têm três características principais: aprendizado, generalização e adaptabilidade. O termo aprendizado é dado porque a rede tem a capacidade de armazenar conhecimento por meio de um processo de ensino. Esse conhecimento é armazenado pelos pesos das conexões entre os neurônios que compõem a rede neural. A capacidade de generalizar significa que resultados razoáveis podem ser obtidos através do uso de dados diferentes daqueles usados durante o processo de treinamento. E o termo adaptabilidade significa que uma rede neural pode ser retreinada online para funcionar adequadamente diante das mudanças em seu ambiente.

Uma rede neural simples, também chamada de perceptron, é composta por um conjunto de neurônios de entrada interligados a um conjunto de neurônios de saída. A Figura 22 apresenta os elementos básicos de uma perceptron, com neurônios de entrada (xi), conectados a um neurônio de saída, onde cada aresta possui um peso (wi), resultando em uma predição y. O neurônio de saída, que recebe como entrada o produto os dados dos neurônios da camada anterior (no caso, da camada de entrada),

e aplica uma função sinal (função de ativação) para converter o produto das entradas para uma faixa restrita.

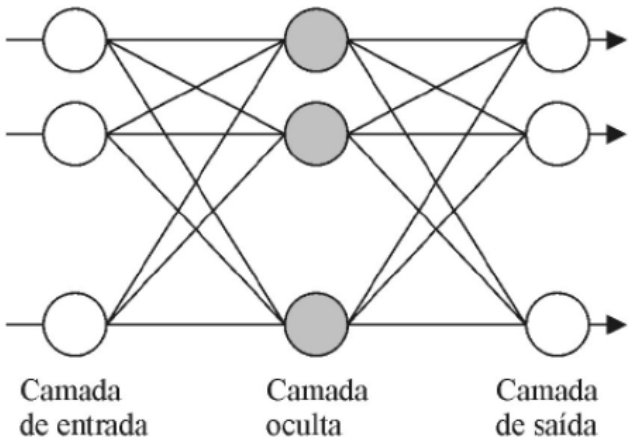
Figura 22 – Arquitetura básica de uma perceptron.



Fonte – O autor.

Uma das arquiteturas de rede neural mais amplamente usadas para reconhecimento de padrões é *Multilayer Perceptron (MLP)*. Consiste em uma arquitetura composta por três componentes principais: uma camada de entrada, uma ou mais camadas ocultas e uma camada de saída (Figura 23). A rede aprende com os erros, durante o treinamento, através da diferença entre a saída desejada e a saída obtida. O treinamento destas redes consiste em duas fases: a primeira é uma fase para frente, na qual as informações colocadas nos neurônios da camada de entrada são propagadas para frente através das camadas ocultas e até a camada de saída, o que gera nos neurônios que a compõem, a resposta ao informações fornecidas como entrada. Durante esta fase, todos os pesos das conexões entre os neurônios são usados; a segunda fase é conhecida como fase reversa, que consiste em modificar os valores de peso de acordo com o erro gerado pelos neurônios na camada de saída.

Figura 23 – MLP.



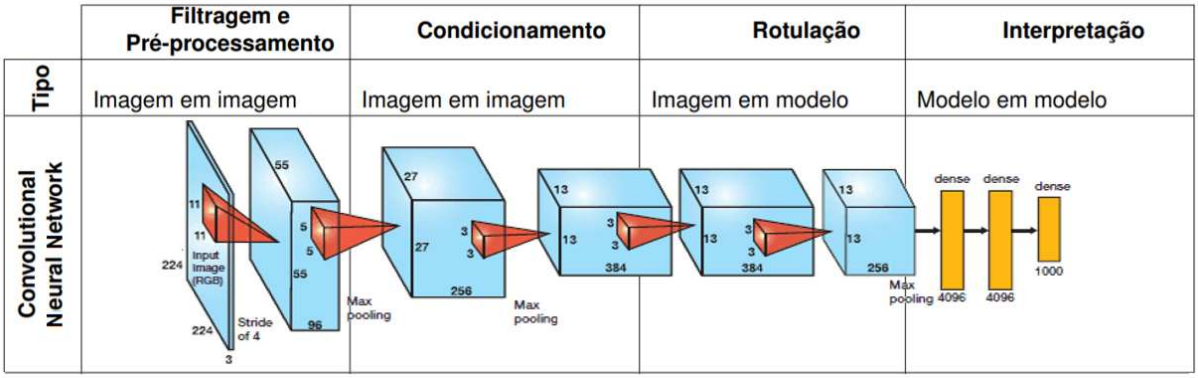
Fonte – (STIVANELLO *et al.*, 2019).

2.3 REDES NEURAIS CONVOLUCIONAIS

Nesta seção, será feita uma breve conceitualização das Redes Neurais Convolucionais. Grande parte da informação foi adquirida de (VON WANGENHEIM, 2020a; YAMASHITA *et al.*, 2018; KARPATHY, 2020a, 2020b, 2020d).

Uma rede neural convolucional ou *CNN* do inglês *Convolutional Neural network* ou *ConvNet* é um tipo especial de rede neural projetada para reconhecer padrões visuais diretamente dos pixels da imagem com pré-processamento mínimo. Elas realizam filtragem, condicionamento, extração de características e classificação de forma integrada. Tudo em um único objeto monolítico (Figura 24).

Figura 24 – Processamento de imagens com CNN.



Fonte – Adaptado de (VON WANGENHEIM, 2020a).

A partir das *CNNs*, é possível solucionar algumas desvantagens presentes nos algoritmos clássicos, uma delas está no processo de extração de características. A desvantagem dos algoritmos clássicos é que esse processo é realizado estaticamente e depende fortemente de algoritmos genéricos. Esses algoritmos podem ser complexos e, para fazer uma boa descrição do conteúdo das imagens, eles devem extrair algumas características das imagens ou partes delas a partir de uma simplificação, por exemplo: segmentação. Os extratores de características não são adaptáveis, são algoritmos projetados para funcionar de maneiras específicas. Se esses extratores não forem escolhidos corretamente, eles poderão não conseguir distinguir adequadamente as diferentes classes de imagens em um problema específico e afetar as outras etapas como filtros, simplificadores e classificadores. Nenhum classificador pode classificar corretamente as imagens que foram descritas com base em um conjunto de descritores que não são relevantes para as classes.

A maneira como as *CNNs* resolvem esse problema é graças aos seus componentes, um dos quais são as camadas convolucionais, que aprendem os descritores das características específicas e personalizadas das imagens durante o treinamento da rede, sem depender de algoritmos genéricos. Além disso, todo o processo, da in-

terpretação à tomada de decisão, é integrado, permitindo que todas as etapas sejam relacionadas. Dessa forma, uma *CNN* com a arquitetura correta e que foi devidamente treinada substitui toda uma sequência de etapas de processamento de imagem (VON WANGENHEIM, 2020a).

2.3.1 Qual é a diferença entre as redes neurais comuns e CNNs?

As *CNNs* são muito semelhantes às redes neurais comuns. São constituídas por neurônios que possuem pesos e bias. Cada neurônio recebe algumas entradas, produz um produto escalar e, opcionalmente, o segue com não linearidade. A diferença é que, nas arquiteturas das *CNN*, as entradas são assumidas explicitamente como imagens ou dados do tipo em grade, o que permite codificar certas propriedades na arquitetura. Isso torna a função de encaminhamento mais eficiente para implementar e reduz bastante o número de parâmetros na rede (KARPATHY, 2020a).

As redes neurais tradicionais recebem uma entrada, um único vetor que é transformado através de uma série de camadas ocultas. Cada camada oculta é composta de um conjunto de neurônios, onde cada neurônio está completamente conectado a todos os neurônios da camada anterior e onde os neurônios de uma única camada trabalham de forma totalmente independente e não compartilham nenhuma conexão. Essas redes não podem ser facilmente escalonadas para imagens complexas, por exemplo, uma imagem de tamanho 32x32x3, deve ter um neurônio totalmente conectado em uma primeira camada oculta com $32 * 32 * 3 = 3072$ pesos, isso significa que imagens maiores aumentarão para um número excessivo de pesos que serão difíceis de manejar, tornando sua conectividade um desperdício.

Ao contrário de uma rede neural normal, as camadas de uma *CNN* possuem neurônios organizados em 3 dimensões: largura, altura e profundidade. A profundidade refere-se à terceira dimensão de um volume de ativação. Os neurônios em uma camada serão conectados apenas a uma pequena região da camada anterior, em vez de todos os neurônios de uma maneira totalmente conectada. Além disso, na camada de saída, a imagem inteira será reduzida por um único vetor de pontuações de classe.

2.3.2 Componentes de uma CNN

Uma CNN é uma construção matemática que geralmente consiste em três tipos de camadas: convolução, agrupamento e camadas totalmente conectadas. As duas primeiras camadas realizam a extração de características, enquanto a terceira, mapeia as características extraídas na saída final (YAMASHITA *et al.*, 2018).

2.3.2.1 Camada de convolução *(CONV)*

Uma camada de convolução é um componente fundamental da arquitetura da *CNN*, é responsável pela extração de caraterísticas, geralmente consistindo em uma combinação de operações lineares e não lineares, ou seja, operações de convolução e funções de ativação. Os parâmetros da camada *CONV* consistem em um conjunto de filtros ou *kernels* que podem ser aprendidos. Cada filtro é espacialmente pequeno em largura e altura, mas se estende por toda a profundidade do volume de entrada. Por exemplo, um filtro típico em uma primeira camada de uma *CNN* pode ter tamanho de 5x5x3, 5 pixels de largura e altura e 3 porque as imagens têm profundidade 3, os canais de cores.

Durante a propagação direta, cada filtro é deslizado ou convolucionado pela largura e altura do volume de entrada, e os produtos ponto entre o filtro e a entrada são calculados. À medida que o filtro desliza, é produzido um mapa de ativação bidimensional que fornece as respostas desse filtro em cada posição espacial, criando um mapa de características. Intuitivamente, a rede aprenderá filtros que serão ativados quando virem algum tipo de característica visual, como uma borda de uma determinada orientação ou um ponto de alguma cor na primeira camada, ou eventualmente reconhecer padrões nas camadas mais altas da rede. Um conjunto de filtros é gerado para cada camada *CONV*, na qual cada um deles produzirá um mapa de ativação bidimensional separado.

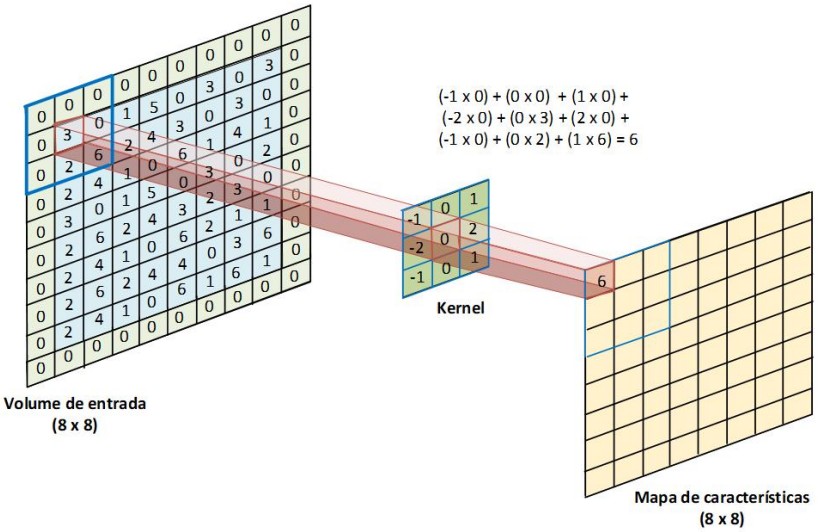
No processo de treinamento da rede, os filtros são os únicos parâmetros que são aprendidos automaticamente; por outro lado, o tamanho do filtro, o número de filtros, um *stride* e um *padding* devem ser estabelecidos antes do início do processo de treinamento; estes são considerados como hiperparâmetros:

a. O *stride*: define o passo com que o filtro é deslizado. Quando o *stride* é 1, os filtros são movidos um pixel de cada vez. Quando o *stride* é 2 os filtros saltam 2 pixels por vez enquanto deslizam. Isso produzirá volumes de saída menores espacialmente.

b. O *padding*: é uma técnica que consiste em adicionar linhas e colunas de zeros a cada lado do volume de entrada, para ajustar o centro de um núcleo no elemento mais externo e manter a mesma dimensão através da operação de convolução (Figura 25).

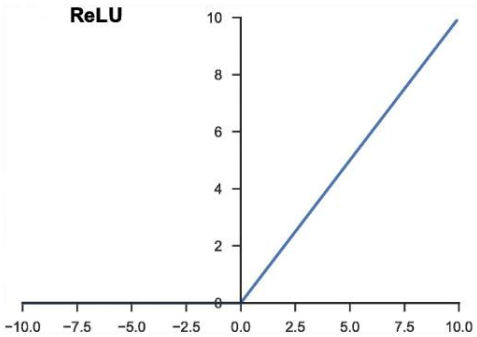
As saídas de uma operação linear, como convolução, são passadas por uma função de ativação não linear. A função comumente usada é a *Rectified Linear Unit (ReLU)*, que simplesmente calcula a função *f(x) = max(0, x)* (Figura 26).

Figura 25 – Operação de convolução com *padding* zero para manter as dimensões no plano.



Fonte – Adaptado de (VON WANGENHEIM, 2019b).

Figura 26 – Função ReLU.

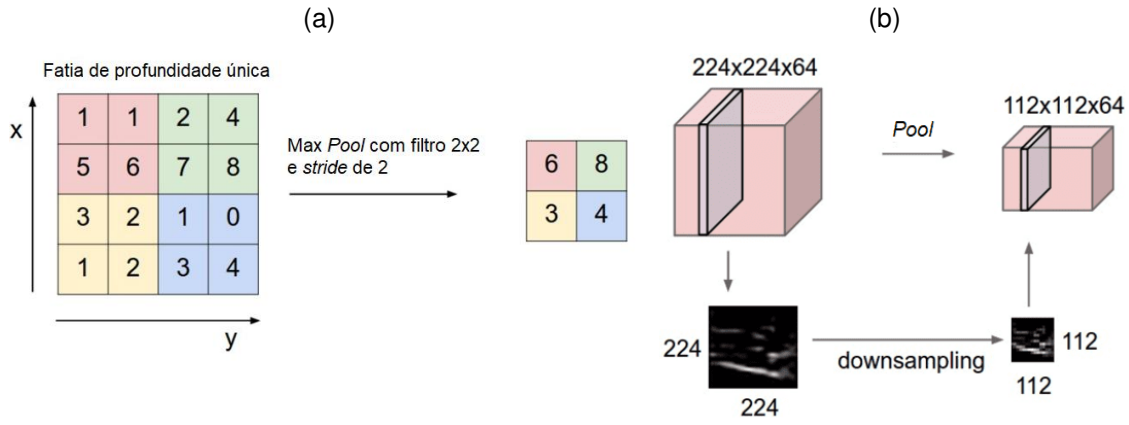


Fonte – (YAMASHITA *et al.*, 2018).

2.3.2.2 Camada de agrupamento *(Pool)*

É comum adicionar periodicamente uma camada de *Pool* entre as camadas sucessivas *CONV* em uma arquitetura *CNN*. Sua função é reduzir progressivamente o tamanho espacial da representação para reduzir o número de parâmetros e o cálculo na rede, controlando o sobreajuste (*overfit*). Na prática, normalmente é usado um agrupamento máximo (max pooling) com um filtro de tamanho 2 × 2 com *stride* de 2 (Figura 27a). Isso permite reduzir a dimensão no plano dos mapas de características em um fator de 2. Ao contrário da altura e largura, a dimensão em profundidade dos mapas de características permanece inalterada (Figura 27b). Além do agrupamento máximo, as unidades de agrupamento também podem executar outras funções, como o agrupamento médio ou o agrupamento padrão *L2* (YAMASHITA *et al.*, 2018).

Figura 27 – Exemplo de *Pool*.



Fonte – Adaptado de (KARPATHY, 2020a).

2.3.2.3 Camada totalmente conectada *(Fully-connected layer (FC))*

Os mapas de caraterísticas de saída da camada final de convolução ou agrupamento são geralmente achatados, ou seja, transformados em uma matriz unidimensional *(1D)* de números ou vetores conectados a uma ou mais camadas totalmente conectadas, também conhecidas como camadas densas, nas quais cada entrada é conectada a cada saída por um peso que pode ser aprendido. A camada final totalmente conectada geralmente possui o mesmo número de nós de saída que o número de classes. Cada *FC* é seguida por uma função não linear, como *ReLU* (YAMASHITA *et al.*, 2018). A função de ativação aplicada à última camada totalmente conectada geralmente é diferente das outras e é selecionada de acordo com cada tarefa. Uma função de ativação aplicada à tarefa de classificação de várias classes é a função *Softmax* que será detalhada um pouco mais no processo de classificação (YAMASHITA *et al.*, 2018).

2.3.3 Treinando uma rede

O treinamento de uma rede é um processo de busca de filtros em camadas *CONV* e pesos em camadas *FC* que minimizam o erro entre previsões de saída e as classes corretas em um conjunto de dados de treinamento. Em redes neurais a função de perda e o algoritmo de otimização de descida do gradiente desempenham papéis essenciais (YAMASHITA *et al.*, 2018).

2.3.3.1 Descida do Gradiente *(Gradient descent)*

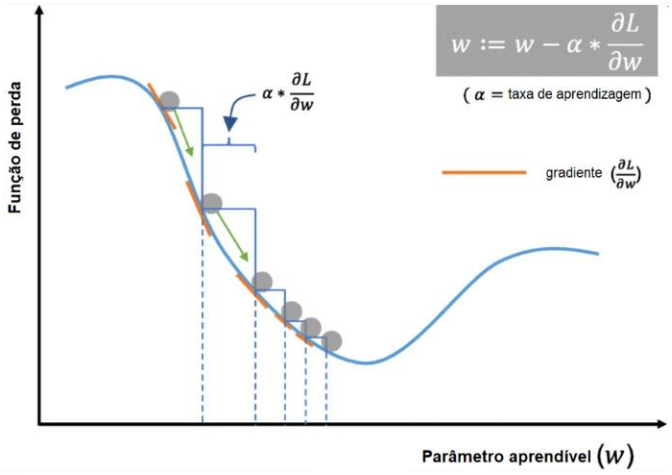
A descida do gradiente é usada como um algoritmo de otimização que atualiza iterativamente os parâmetros que podem ser aprendidos: filtros e pesos, para minimizar a perda. O gradiente da função de perda fornece a direção na qual a função tem a

maior taxa de aumento e cada parâmetro é atualizado na direção negativa do gradiente com um tamanho de passo arbitrário determinado por um hiperparâmetro chamado taxa de aprendizado (*learning rate*) (Figura 28). O gradiente é, matematicamente, uma derivada parcial da função de perda em relação a cada parâmetro que pode ser aprendido. Uma atualização de um parâmetro é formulada da seguinte maneira:

$$w := w - \alpha * \frac{\partial L}{\partial w} \tag{4}$$

Onde w representa cada parâmetro que pode ser aprendido, α é uma taxa de aprendizado e L uma função de perda.

Figura 28 – Descida do gradiente.



Fonte – Adaptado de (YAMASHITA *et al.*, 2018).

Devido às limitações de memória, os gradientes da função de perda em relação aos parâmetros são calculados usando um subconjunto do conjunto de dados de treinamento conhecido como mini-lote. Esse método é chamado descida de gradiente de mini-lote, também conhecido como *Stochastic Gradient Descent (SGD)*, onde o tamanho do mini-lote é um hiperparâmetro. Muitas melhorias no algoritmo de descida de gradiente foram propostas e amplamente utilizadas, como: *SGD with momentum, RMSprop, e Adam* (QIAN, 1999; KINGMA; BA, 2014; RUDER, 2016).

2.3.3.2 Dados e rótulos reais *(Data and ground truth labels)*

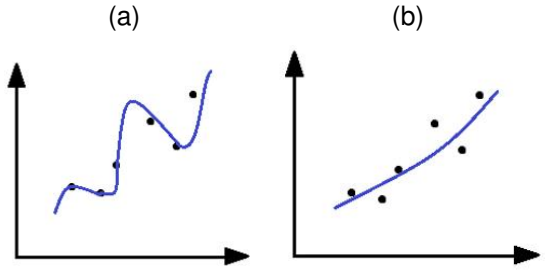
Os dados são os componentes mais importantes para o aprendizado profundo ou outros métodos de aprendizado de máquina. Os dados são divididos em três conjuntos: treinamento, validação e conjunto de testes. O conjunto de treinamento é usado para treinar a rede. O conjunto de validação é usado para avaliar o modelo durante o processo de treinamento, ajustar os hiperparâmetros e fazer a seleção do modelo. O conjunto de testes é usado apenas uma vez no final do processo para avaliar o desempenho do modelo final. Por outro lado, os rótulos reais são os resultados ideais

esperados no processo, neste caso será definido como a classe correta esperada na classificação.

2.3.3.3 *Overfit*

O *Overfit* é gerado quando um modelo aprende casos particulares do conjunto de treinamento. Um exemplo é mostrado na Figura 29a, em que o modelo acaba memorizando ruídos irrelevantes em vez de aprender o sinal, como é o caso correto do modelo mostrado na Figura 29b.

Figura 29 – (a) Modelo com *overfit* e (b) Modelo correto.



Fonte – O autor.

O *overfit* é um dos principais desafios do aprendizado de máquina, uma vez que um modelo com *overfit* não pode ser generalizado; por isso, uma série de técnicas foram propostas para ajudar a mitigá-lo:

a. Mais dados de treinamento: a melhor solução para reduzir o *overfit* é obter mais dados de treinamento. Um modelo treinado em um conjunto de dados maior geralmente generaliza melhor.

b. Aumento de dados: é um processo de modificação de dados de treinamento por meio de transformações aleatórias, como inversão, translação, corte, rotação e exclusão aleatória para gerar variabilidade nas imagens de treinamento (ZHONG *et al.*, 2017).

c. Regularização com *dropout* ou *weight decay*: o *dropout* é uma técnica de regularização na qual algumas ativações selecionadas aleatoriamente são definidas como zero durante o treinamento, para que o modelo se torne menos sensível a pesos específicos na rede (GILL *et al.*, 2020). O *weight decay*, também conhecido como regularização *L2*, reduz o excesso de ajuste penalizando a magnitude do modelo, para que os pesos obtenham apenas valores pequenos.

d. Normalização por lotes: é um tipo de camada suplementar que normaliza adaptativamente os valores de entrada da próxima camada, atenua o risco de *overfit*,

além de melhorar o fluxo de gradiente pela rede, permite maiores taxas de aprendizado e reduz a dependência da inicialização (IOFFE; SZEGEDY, 2015).

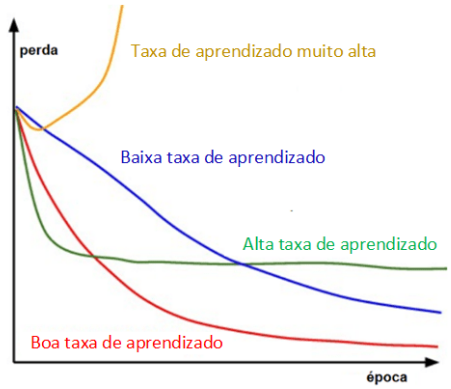
2.3.4 Monitorização do processo de aprendizagem

Existem várias quantidades úteis para monitorar durante o treinamento da rede neural. Os gráficos mostrados abaixo permitem visualizar o comportamento das diferentes configurações de hiperparâmetros durante o processo de treinamento. O eixo *x* dos gráficos estão em unidades de época, que medem quantas vezes todos os exemplos foram visualizados durante o treinamento. É preferível rastrear épocas em vez de iterações, pois o número de iterações depende da configuração arbitrária do tamanho do lote (KARPATHY, 2020c).

2.3.4.1 Função de perda

A primeira quantidade útil para rastrear durante o treinamento é a perda. A Figura 30 mostra um diagrama que relaciona a perda com o tempo. A imagem representa os efeitos de diferentes taxas de aprendizado. Com baixas taxas de aprendizado (linha azul), as melhorias serão lineares. Com taxas de aprendizado muito altas (linha amarela), há uma divergência de aprendizagem. Com altas taxas de aprendizado (linha verde) diminuirão a perda mais rapidamente, mas ficam presas nos piores valores de perda. Isso ocorre porque há muita “energia” na otimização e os parâmetros estão flutuando caoticamente, incapazes de se estabelecer em um local adequado no cenário da otimização (KARPATHY, 2020c).

Figura 30 – Relação da perda com a taxa de aprendizado.



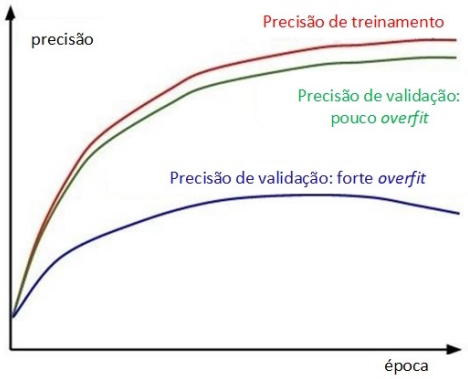
Fonte – Adaptado de (KARPATHY, 2020c).

2.3.4.2 Precisão do treinamento/validação

A precisão do treinamento/validação é a segunda quantia importante a seguir durante o treinamento. Este diagrama pode fornecer informações valiosas sobre a quantidade de *overfit* no modelo. Dois casos possíveis são mostrados na Figura 31.

A curva de erro de validação azul mostra muita pouca precisão de validação em comparação com a precisão do treinamento, indicando um forte *overfit* onde a precisão de validação pode até começar a diminuir após algum ponto. Para resolver isso, é necessário aumentar a regularização. O outro caso é quando a precisão da validação segue muito bem a precisão do treinamento. Este caso indica que a precisão é baixa, e é necessário aumentar o numero de parâmetros.

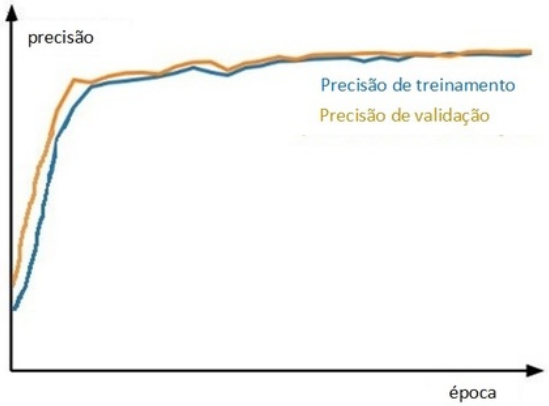
Figura 31 – Diagrama da relação entre precisão de treinamento/validação.



Fonte – Adaptado de (KARPATHY, 2020c).

Por outro lado, um bom ajuste é identificado quando a precisão do treinamento e da validação aumentam até um ponto de estabilidade com um intervalo mínimo entre os dois valores finais. A precisão quase sempre será maior no conjunto de dados de treinamento do que no conjunto de dados de validação, o que significa que um certo intervalo deve ser esperado entre as duas curvas (BROWNLEE, 2019). O diagrama na Figura 32 mostra um caso de um bom ajuste.

Figura 32 – Exemplo de um bom ajuste entre a precisão de treinamento e a precisão de validação.



Fonte – Adaptado de (BROWNLEE, 2019).

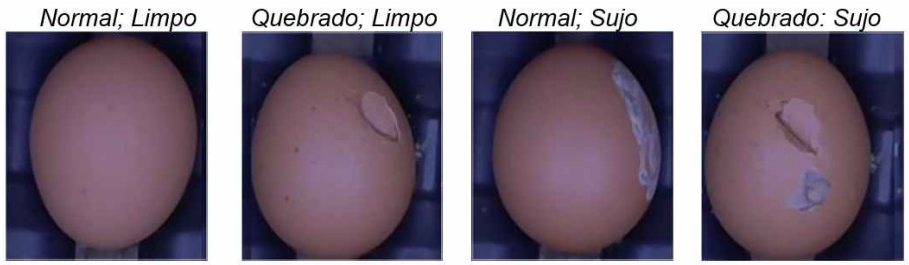
2.3.5 Funções de uma CNN

CNNs podem ser usados para vários fins, como: classificar imagens, classificar cada pixel de uma imagem com uma determinada classe, conhecido como segmentação semântica, detectar objetos em imagens com sua localização e detectar e segmentar ao mesmo tempo. Este projeto visa a classificação, onde tanto a classificação de imagens por reconhecimento de objetos quanto a segmentação semântica podem ser úteis.

2.3.5.1 Classificação de imagem por reconhecimento de objetos

Classificar uma imagem é atribuir uma ou mais classes a uma imagem de entrada de um conjunto fixo de classes. O processo de atribuição de várias classes a uma imagem é conhecido como classificação de rótulos múltiplos *(multi-tag)*. Um exemplo é mostrado na Figura 33, onde o ovo é classificado como *normal* ou *quebrado* e como *sujo* ou *limpo*.

Figura 33 – Classificação *multi-tag*.



Fonte – O autor.

Para classificar uma imagem, dois componentes principais devem ser considerados: uma função de pontuação, que mapeia os valores de pixel de uma imagem para as pontuações de confiança de cada classe, e uma função de perda, que quantifica a concordância entre as pontuações previstas e as classes corretas (KARPATHY, 2020d). O primeiro passo é definir a função de pontuação. Possivelmente, a função mais simples para obter esse mapeamento é através de um mapeamento linear.

Um classificador linear possui dois componentes: o primeiro é a imagem geralmente descrita por x representando os dados de entrada, e o segundo é um conjunto de parâmetros ou pesos normalmente descritos com a letra W. Cada x_i é associada a uma classe y_i onde $i = 1...N$ e $y_i \in 1...K$. N é o número de exemplos cada um com uma dimensão D e K as diferentes classes. Os parâmetros x e W são unidos a partir de uma função f:

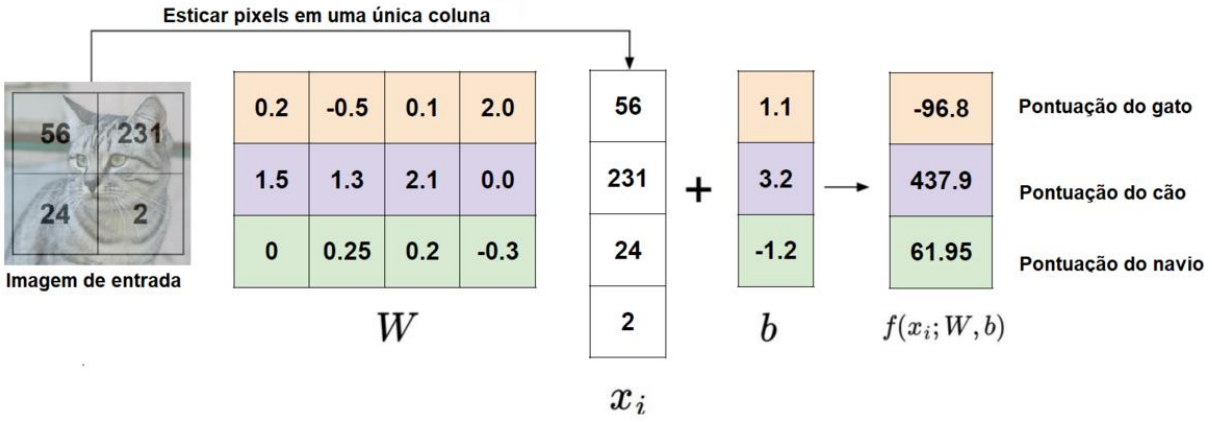
$$f(x_i, W, b) = Wx_i + b \tag{5}$$

Onde x_i tem todos os seus pixels achatados em um único vetor de coluna da forma *[D x 1]*. A matriz W de tamanho *[K x D]* e o vetor b de tamanho *[K x 1]*. O

parâmetro *b* é conhecido como vetor de viés porque influencia as pontuações de saída, mas sem interagir com os dados reais x_i.

O classificador linear calcula a pontuação de uma classe como uma soma ponderada de todos os seus valores de pixel nos 3 canais de cores. Dependendo da precisão dos valores estabelecidos para esses pesos, a função tem a capacidade de diferenciar determinadas cores em determinadas posições da imagem. Na Figura 34, é apresentado um exemplo de mapeamento de uma imagem para pontuação de classe. Para facilitar a explicação, é assumido que a imagem tem apenas 4 pixels monocromáticos, os canais de cores não são considerados por simplicidade e 3 classes: gato (amarelo), cão (roxa) e navio (verde). Os pixels da imagem são esticados em uma coluna e a multiplicação da matriz é realizada para obter as pontuações para cada classe. O conjunto de pesos *W* foi escolhido aleatoriamente; nesse caso, é obtido um cão com a pontuação mais alta como resposta.

Figura 34 – Mapeando uma imagem para as pontuações da classe.



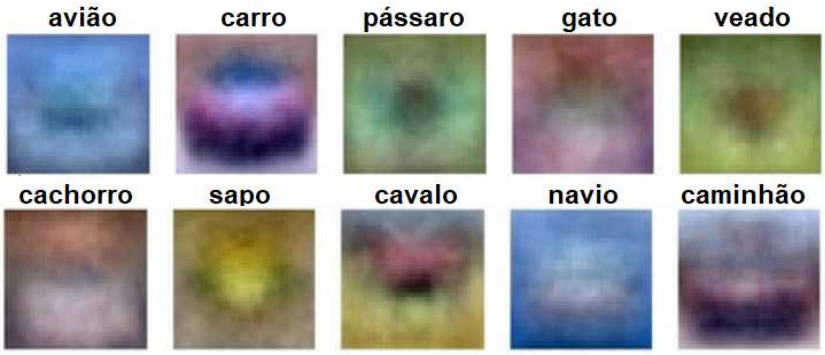
Fonte – Adaptado de (STANFORD, 2017b).

Uma maneira de interpretar os pesos *W* é que cada linha de *W* corresponde a um modelo para uma das classes. A pontuação para cada classe de uma imagem é obtida comparando cada modelo com a imagem usando um produto interno ou produto pontual um por um para encontrar o que melhor se encaixa. A Figura 35 mostra um exemplo de modelos obtidos no final de um treinamento com *CIFAR-10* (KRIZHEVSKY *et al.*, 2009). Pode-se observar que o modelo *"navio"* contém muitos pixels azuis. Portanto, este modelo dará uma pontuação alta toda vez que for comparado com imagens de navios no oceano com um produto interno.

O classificador linear tem algumas desvantagens, uma delas é que a combinação de várias imagens gera distorções dos objetos nos modelos finais. Um exemplo dessa distorção é observado no modelo associado ao *"cavalo"* na Figura 35, onde o modelo acabou sendo um cavalo com duas cabeças, esse resultado foi obtido porque no conjunto de treinamento havia imagens de cavalos orientados nas duas direções:

esquerda e direita. Outra desvantagem é observada no modelo associado ao *"carro"*, onde o modelo final acabou sendo vermelho, o que sugere que nos dados de treinamento havia mais carros vermelhos do que qualquer outra cor. Por esse motivo, o classificador será muito fraco para diferenciar e classificar carros de cores diferentes.

Figura 35 – Exemplo de pesos aprendidos no final do aprendizado para *CIFAR-10*.



Fonte – Adaptado de (KARPATHY, 2020b).

Uma rede neural pode lidar com esses tipos de problemas, pois pode desenvolver neurônios intermediários em suas camadas ocultas que detectam tipos específicos de carros. Por exemplo, um carro verde à esquerda, um carro azul na frente. Os neurônios na próxima camada poderiam combiná-los em uma pontuação de carro mais precisa, usando uma soma ponderada de detectores de carros individuais (KARPATHY, 2020b).

Uma vez que a função de pontuação é definida, a próxima etapa é definir a função de perda *(Loss function)*. Considerando a Figura 34, verificou-se que o conjunto específico de pesos nesse exemplo não foi muito bom, os pixels na imagem de entrada representavam um gato, mas a pontuação do gato saiu muito baixa (-96.8) em comparação com outras classes. Nesta etapa, a infelicidade desses resultados será medida, com base em uma função de perda. A perda será alta quando o classificador não obtiver bons resultados. Há várias maneiras de definir os detalhes da função de perda entre as mais comuns estão:

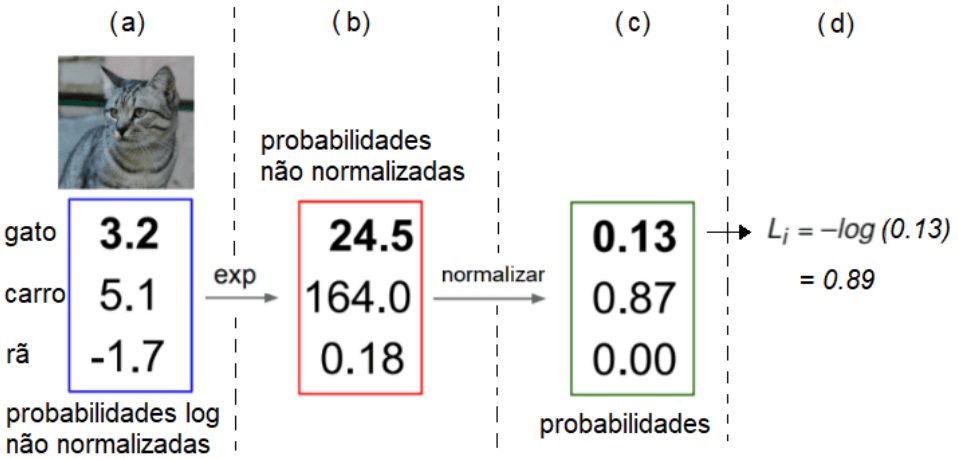
a. *Multinomial Logistic Regression (Softmax):* Tem a característica de definir as saídas f(x_i, W) de maneira intuitiva como probabilidades de classe normalizadas. O *Softmax* tem a seguinte forma (KARPATHY, 2020d):

$$L_i = -log\left(\frac{e^{f_{yi}}}{\sum_j e^{f_i}}\right) \tag{6}$$

onde a notação f_j está sendo usada para significar o j-ésimo elemento do vetor de pontuação da classe *f*.

O *Softmax* interpreta as pontuações do vetor de saída *f* como probabilidades de *log* não normalizadas (Figura 36a). Encontrando o exponencial desses resultados não normalizados (Figura 36b) e dividindo pela soma das probabilidades *log* não normalizadas, obtém-se um vetor de pontuações em que as probabilidades somam um (Figura 36c). Como o objetivo da função *Softmax* é tornar a resposta verdadeira o mais próximo possível de 1, uma função monotônica como *(log)* é usada, porque é matematicamente mais fácil maximizar, no entanto, como as funções de perda medem o mal e não o bem, é por isso que a função de *log* negativo é usada para direcionar a resposta na direção certa. Obtendo o resultado mostrado na Figura 36d.

Figura 36 – Exemplo concreto mostrando as etapas para encontrar as pseudo probabilidades associadas a cada classe usando o classificador *Softmax*.



Fonte – Adaptado de (STANFORD, 2017c).

b. *Perda de entropia cruzada (Cross-entropy loss)*: esta função de perda é amplamente utilizada, é rápida, confiável e funciona perfeitamente quando se tem um ou mais de dois classes na saída. Esta função aplica o conceito de probabilidades logarítmicas usadas por a função *Softmax* junto a função *Negative Log Likelihood (NLL)*, que basicamente seleciona a perda negativa onde a classe é verdadeira. Para entender como o NLL funciona, um exemplo é apresentado na Tabela 1, as colunas 2 e 3 mostram o resultado das probabilidades logarítmicas obtidas usando *Softmax* para 5 entradas com duas classes (0 e 1), na coluna 4 é apresentada a classe correta de cada entrada e na coluna 5 é mostrado o resultado da perda escolhida. Como pode ser visto, apenas a ativação da coluna que contém a classe correta é selecionada, com isso se tem uma função de perda que mostra o quão bem cada classe está sendo prevista. Não é necessário considerar as demais colunas, pois conforme definido pela função *Softmax*, todas as ativações somam 1 menos a ativação correspondente à classe correta. Portanto,

tornar a ativação da classe correta o mais alto possível significa que também está-se diminuindo a ativação das colunas restantes. *NLL* é basicamente este processo, exceto que considera o negativo [-0.6024, -0.4979, . . .]. Finalmente, tomando a média das perdas, obtém-se a perda de entropia cruzada.

Tabela 1 – Exemplo de uso de *Softmax* com *NLL*.

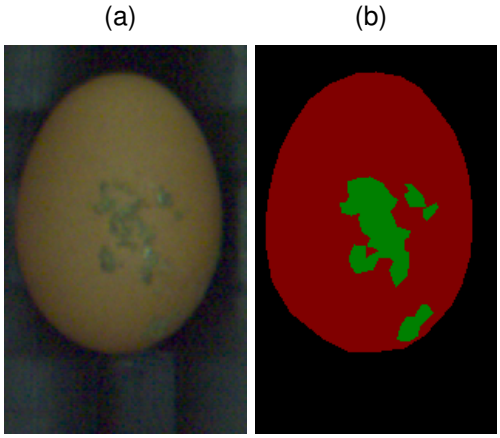
Entrada	***Softmax***		**Classe correta**	**Perda**
	Classe 0	**Classe 1**		
0	0.602469	0.397531	0	0.602469
1	0.502065	0.497935	1	0.497935
2	0.133188	0.866811	0	0.133188
3	0.99664	0.003361	1	0.003361
4	0.595949	0.404051	1	0.404051

Fonte – Adaptado de (HOWARD, 2020a).

2.3.5.2 Segmentação semântica

Neste processo, uma imagem é inserida e uma classe é obtida para cada pixel na imagem. Por exemplo, no caso da Figura 37, espera-se que cada pixel da imagem receba uma classe: fundo, ovo, sujeira ou algum outro tipo de classe predefinida.

Figura 37 – (a) Imagem de entrada (b) Imagem de saída, fundo: preto, ovo: vermelho e sujeira: verde.

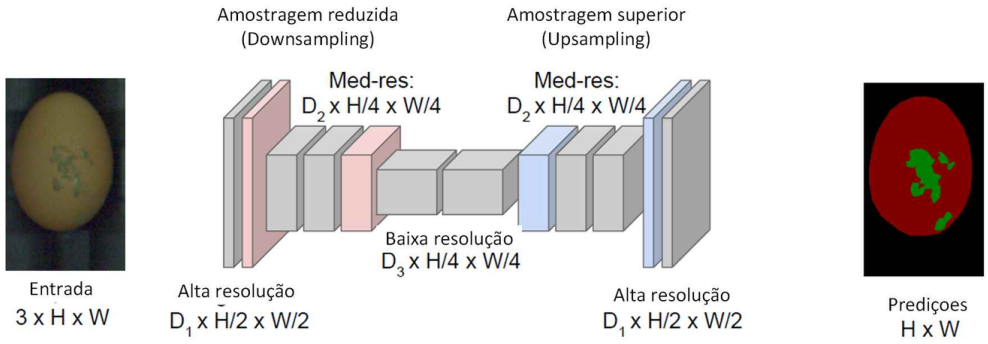


Fonte – O autor.

Este processo é computacionalmente caro se apenas *CNNs* simples forem usadas, visto que uma imagem com a mesma resolução da imagem de entrada deve ser obtida como saída. Isso implicaria em realizar várias operações de convolução com grandes filtros que consumiriam muita memória. É por isso que a rede utilizada para este processo é dividida em duas partes, uma primeira parte para amostragem

reduzida (*Downsampling*) e uma segunda parte para amostragem ascendente (*Upsampling*) (Figura 38). As arquiteturas CNN simples são usadas na primeira parte da rede para produzir representações abstratas da imagem de entrada. Na segunda parte da rede as representações de imagens abstratas são obtidas usando várias técnicas que permitem que as dimensões da imagem de entrada sejam adquiridas novamente, este tipo de rede é conhecida codificador-decodificador.

Figura 38 – Rede codificador-decodificador.



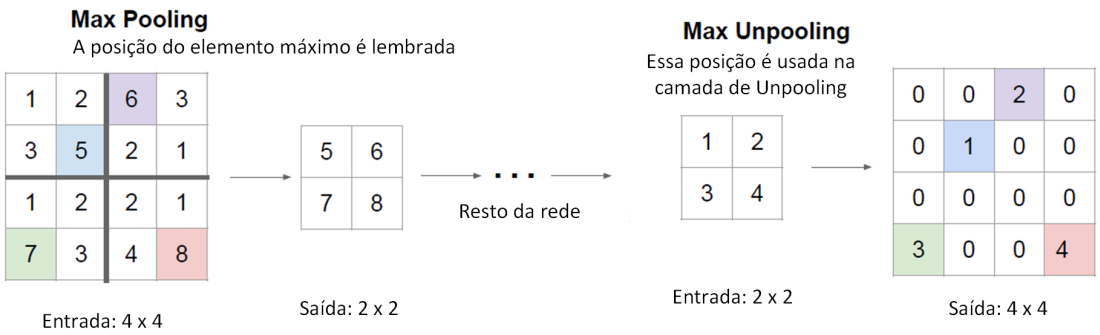
Fonte – Adaptado de (STANFORD, 2017a).

Entre as técnicas mais utilizadas para amostragem ascendente neste tipo de rede estão: o vizinho mais próximo (Figura 39a) que pega um valor de pixel de entrada e é copiado para os vizinhos K mais próximos, onde K depende da saída esperada, *Bed of Nails* (Figura 39b) onde o valor do pixel de entrada é copiado na primeira posição da imagem de saída e as posições restantes são preenchidas com zeros, e o *MaxUnpooling* (Figura 40) onde o índice do valor máximo de cada camada de agrupamento é salvo durante a etapa de codificação, este índice é usado novamente na etapa de decodificação, onde o pixel de entrada é atribuído ao índice salvo, preenchendo as outras posições com zeros.

Figura 39 – Técnicas de amostragem ascendente.

Fonte – Adaptado de (STANFORD, 2017a).

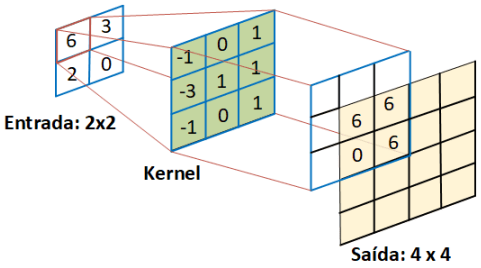
Figura 40 – *MaxUnpooling*



Fonte – Adaptado de (STANFORD, 2017a).

Como convoluções são realizadas na primeira parte da rede, é necessário realizar convoluções transpostas no processo de decodificação, essas convoluções possibilitam aumentar a amostra do mapa de características de entrada para um mapa de características de saída. Para explicar como funciona, é considerado um mapa de características 2x2 que será amostrado em um mapa de características 4x4 (Figura 41), a convolução transposta funciona um pouco diferente da convolução normal, neste caso, em vez de fazer um produto interno entre a entrada e o kernel, o que é feito é pegar cada valor do mapa de características de entrada como um escalar e multiplicá-lo pelos valores do kernel, esses valores são copiados para o mapa de características de saída. Este processo é realizado para todos os valores do mapa de características de entrada, em caso de sobreposição entre os valores são simplesmente somados.

Figura 41 – Processo de convoluções transpostas.



Fonte – O autor.

2.3.6 Arquiteturas

Existem várias arquiteturas no campo das redes convolucionais. Os mais comuns são (KARPATHY, 2020a):

a. *LeNet* (LECUN *et al.*, 1998): desenvolvida por Yann LeCun, essa arquitetura foi uma das primeiras aplicações bem-sucedidas das redes convolucionais da década de 1990. Foi usada para ler códigos postais, dígitos, etc.

b. *AlexNet* (KRIZHEVSKY *et al.*, 2012): desenvolvida por Alex Krizhevsky, Ilya Sutskever e Geoff Hinton. *AlexNet* foi o primeiro trabalho que popularizou as redes convolucionais em visão artificial. Essa rede apresenta uma arquitetura muito semelhante à *LeNet*, porém é mais profunda, maior e apresenta camadas convolucionais empilhadas umas sobre as outras, anteriormente era comum ter uma única camada *CONV* sempre seguida imediatamente por uma camada de *POOL*.

c. *ZFNet* (ZEILER, Matthew D; FERGUS, Rob, 2014): desenvolvida por Matthew Zeiler e Rob Fergus. Tornou-se conhecido como *ZFNet* (abreviação de Zeiler & Fergus Net). Foi uma melhoria do *AlexNet*, ajustando os hiperparâmetros da arquitetura; isso foi alcançado aumentando o tamanho das camadas convolucionais médias e reduzindo o tamanho do *stride* e do filtro na primeira camada.

d. *GoogLeNet* (SZEGEDY *et al.*, 2015): desenvolvida por Szegedy et al. do Google. Sua principal contribuição foi o desenvolvimento de um módulo inicial que reduziu drasticamente o número de parâmetros na rede (4M, comparado ao *AlexNet* com 60M). Além disso, essa arquitetura usa o *Average Pooling* em vez das camadas totalmente conectadas na parte superior da *CNN*, eliminando uma série de parâmetros sem importância. Existem várias versões de rastreamento do *GoogLeNet*, sendo a mais recente a *Inception-v4*.

e. *VGGNet* (SIMONYAN; ZISSERMAN, 2014): desenvolvida por Karen Simonyan e Andrew Zisserman. Sua principal contribuição foi demonstrar que a profundidade da rede é um componente crítico para o bom desempenho. A melhor rede contém 16 camadas *CONV/FC* e apresenta uma arquitetura extremamente homogênea que executa apenas convoluções 3x3 e agrupamento 2x2 do início ao fim. Uma desvantagem do *VGGNet* é que é mais caro avaliar e usa muito mais memória e parâmetros (140M). Esses parâmetros podem ser reduzidos removendo a primeira camada *FC* do modelo sem afetar seu desempenho.

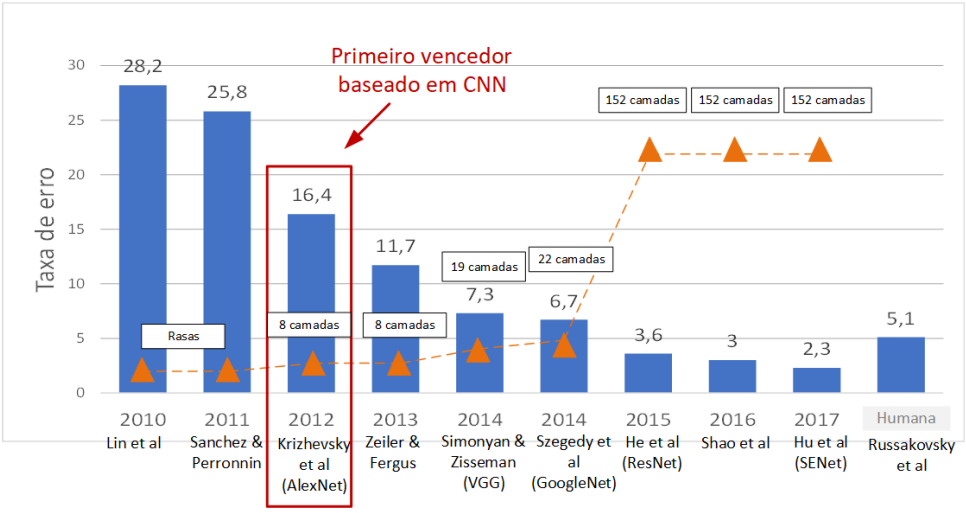
f. *ResNet* (HE *et al.*, 2016): rede residual desenvolvida por Kaiming He et al. Possui conexões especiais de omissão *(Identity Shortcut Connections ou Residual Blocks)* e uso intenso da normalização por lotes. Atualmente, as *ResNets* são modelos de rede neural convolucional de ponta e são a opção padrão para usar *CNNs* na prática (KARPATHY, 2020a). Essa arquitetura consegue treinar uma rede com até 152 camadas, com uma complexidade final menor que a *VGGNet*.

g. *Squeeze-and-Excitation Networks (SENet)* (HU *et al.*, 2018): desenvolvida por Jie Hu et al. É uma melhoria da arquitetura *ResNet*. Essa arquitetura adicionou um módulo de "*feature recalibration*" que recalibra adaptativamente as respostas das características do canal, modelando explicitamente as interdependências entre os canais. Eles demonstraram que o empilhamento de vários blocos conhecidos

como *Squeeze-and-Excitation (SE)* pode criar arquiteturas que se generalizam extremamente bem em conjuntos de dados desafiadores com custo computacional mínimo.

Em 2010, foi criado um desafio de reconhecimento visual conhecido como *ImageNet Large Scale Visual Recognition Challenge (ILSVRC)* (RUSSAKOVSKY *et al.*, 2015); este desafio significou uma importante referência na classificação e detecção de classes de objetos em um banco de dados com milhões de imagens, apesar do fato de que o desafio não é mais realizado desde 2017, é um pilar fundamental para visualizar a trajetória de cada uma das arquiteturas e as diferentes melhorias obtidas ao longo dos anos. Na Figura 42, é apresentada uma comparação das arquiteturas vencedoras do desafio *ILSVRC*, onde é possível ver que em 2012, a arquitetura *AlexNet* foi a primeira a introduzir aprendizado profundo, conseguindo superar outras arquiteturas por uma margem significativa. A partir dessa data, arquiteturas cada vez mais profundas começaram a ser utilizadas, obtendo melhores resultados, como foi o caso da arquitetura *ResNet* que venceu o desafio em 2015, com um total de 152 camadas alcançou uma taxa de erro de 3.57. Finalmente, no último ano, a melhor taxa de erro alcançada foi de 2.3 obtida pela arquitetura *SENet*.

Figura 42 – Vencedores do Desafio *ILSVRC*.

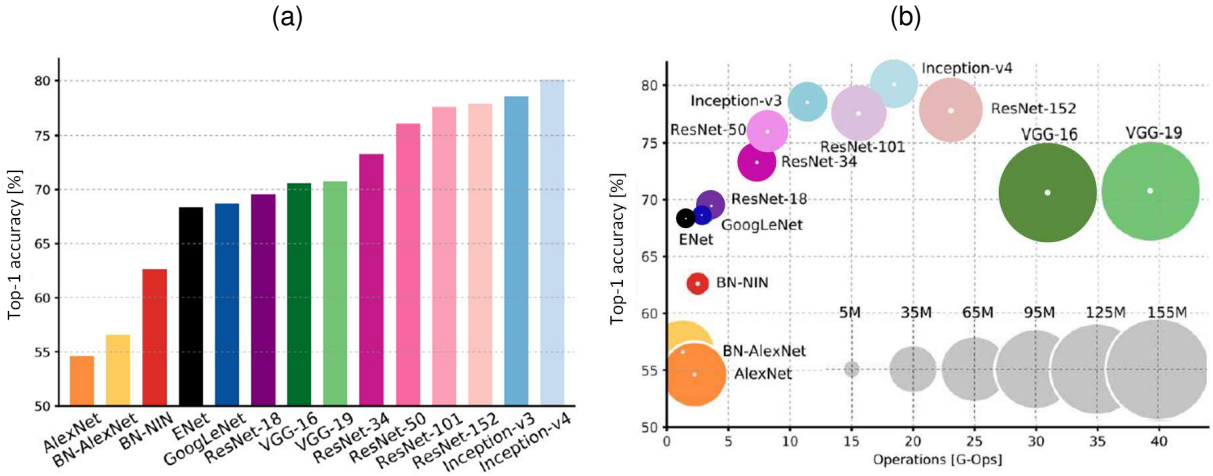


Fonte – (FEI-FEI *et al.*, 2020).

A Figura 43 mostra uma comparação entre a complexidade envolvida nos modelos mencionados e algumas de suas versões. Na Figura 43a, é feita uma comparação de desempenho em termos de precisão, na qual o *GoogLeNet* com sua versão *Inception-V4 (Resnet + inception)* alcançou a melhor precisão entre os outros modelos. Na Figura 43b é feita uma comparação entre a complexidade computacional desses modelos, o eixo *y* representa a precisão, o eixo *x* representa o número de operações utilizadas; portanto, mais à direita representa um maior custo computacional e, o tama-

nho do círculo, representa o uso da memória, ou seja, quanto maior o uso da memória, maior será o círculo. Os círculos cinza são usados como referência. Nesta figura, pode-se observar que os modelos *VGG* (círculo verde) são os modelos menos eficientes, pois ocupam a maior memória e utilizam o maior número de operações, no entanto, possuem boa precisão. O *GoogLeNet* (azul escuro), o mais eficiente nessa comparação, além de um pequeno círculo para uso da memória. No caso do *ResNet*, possui uma eficiência moderada, dependendo do modelo, está no meio, tanto em termos de memória quanto de número de operações e presentam alta precisão.

Figura 43 – Uma análise de modelos de redes neurais profundas para aplicações práticas, 2017.



Fonte – (CANZIANI *et al.*, 2016).

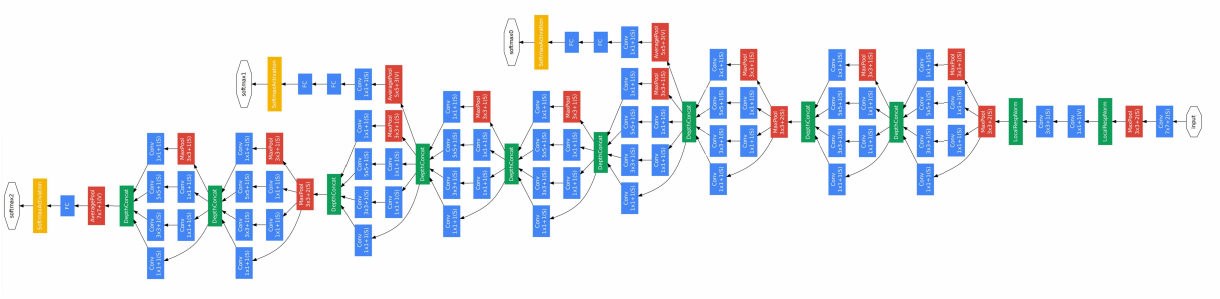
Para este projeto, foram selecionadas as arquiteturas *GoogLeNet* e *ResNet* a serem avaliadas. O *GoogLeNet* foi selecionado por ser uma das arquiteturas mais eficientes de acordo com a Figura 43b em relação ao número de operações e uso de memória. A arquitetura *ResNet* não só tem alta precisão, mas também tem grandes vantagens, uma delas é que permite implementar redes profundas sem arriscar a perda do gradiente quando o erro é retropropagado, e também é possível implementar uma técnica de otimização de hiperparâmetros de entrada, que permite ajustar durante o treinamento a taxa de aprendizado entre uma faixa de valores. Alguns dos componentes principais destas arquiteturas são descritos abaixo:

2.3.6.1 *GoogLeNet*

A arquitetura completa do *GoogLeNet* é apresentada na Figura 44, esta arquitetura usa um modelo CNN inspirado em *LeNet*, e introduz o conceito não sequencial "*Inception module*", este módulo desmonta as camadas individuais e realiza operações de convolução paralelas com diferentes tamanhos nos filtros que são posteriormente

concatenados. Essas pequenas convoluções reduzem drasticamente o número de parâmetros e tornam o treinamento mais fácil. Para evitar que a rede muito profunda morra devido ao esvanecimento de gradiente, os autores introduziram classificadores intermediários. Eles são representados pelos blocos amarelos na Figura 44. Esses classificadores assumem a forma de redes convolucionais menores que permitem que as perdas das camadas intermediárias sejam adicionadas à perda total. *GoogLeNet* tem 9 módulos iniciais empilhados linearmente. Possui 22 camadas de profundidade, 27, incluindo camadas de agrupamento e usa o agrupamento médio global no final do último bloco de inicialização.

Figura 44 – *Arquitetura GoogLeNet.*

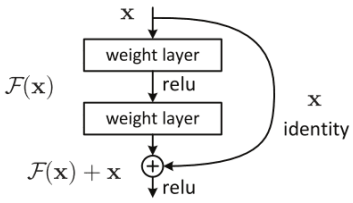


Fonte – (SZEGEDY *et al.*, 2015).

2.3.6.2 *ResNet*

ResNet apresenta uma estrutura de aprendizagem residual para facilitar a formação de redes substancialmente mais profundas a partir do uso de *Identity shortcut connections (residual blocks)*. O objetivo dos *residual blocks* na *ResNet* é impedir que a rede, muito profunda, morra por esvanecimento de gradientes[1]. A *ResNet* usa, em um determinado ponto, um sinal que é a soma do sinal produzido pelas duas camadas convolucionais anteriores mais o sinal transmitido diretamente do ponto anterior para essas camadas, unindo um sinal processado a uma etapa antes do processamento (Figura 45).

Figura 45 – *Identity shortcut connections.*

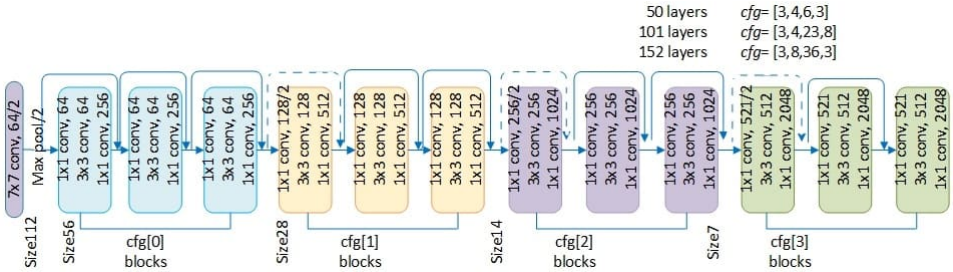


Fonte – (HE *et al.*, 2016).

[1] Perda de erro durante a retropropagação

A ideia geral da arquitetura *ResNet* é uma estrutura de quatro tipos de blocos de camadas, enumerados de 0 a 3, que podem ser usados para construir redes de diferentes profundidades. *ResNet50*, por exemplo, usa uma configuração (3, 4, 6, 3). A Figura 46 mostra uma visão abstrata de uma arquitetura com uma configuração (3, 3, 3, 3).

Figura 46 – Visão abstrata de uma arquitetura *ResNet* com uma configuração (3, 3, 3, 3).



Fonte – (VON WANGENHEIM, 2020b).

A Figura 47 resume os diferentes modelos específicos de *ResNet* descritos na literatura:

Figura 47 – Modelos específicos de *ResNet*

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

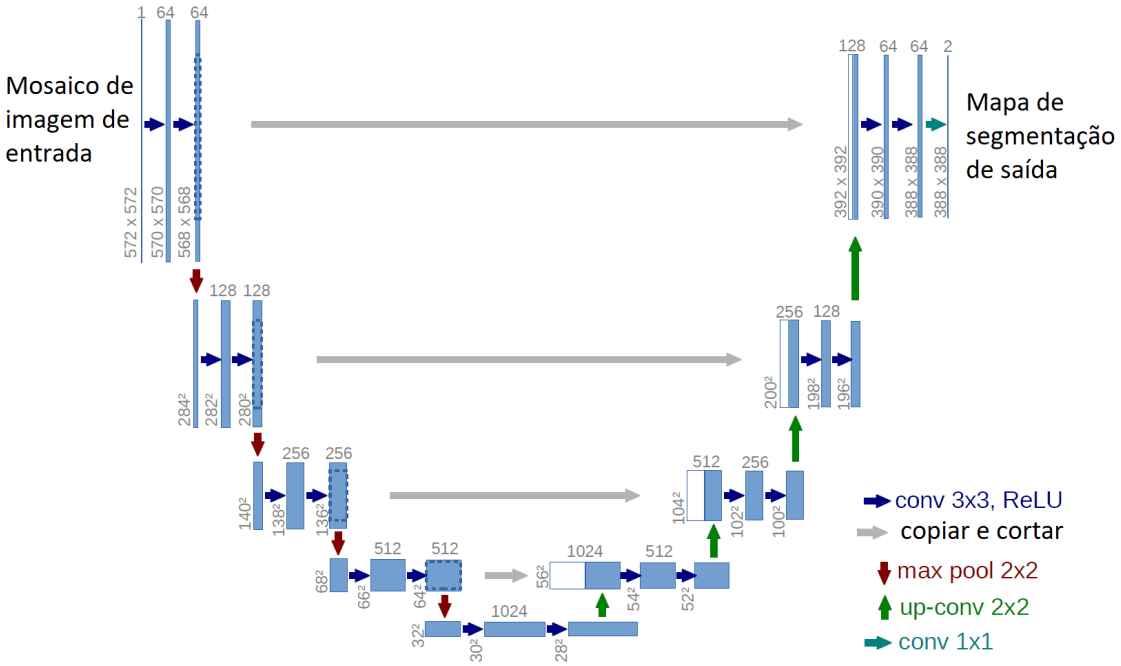
Fonte – (HE *et al.*, 2016).

2.3.6.3 Arquiteturas para segmentação semântica

Entre as arquiteturas mais utilizadas para segmentação semântica estão *Unet* (RONNEBERGER *et al.*, 2015) e *SegNet* (BADRINARAYANAN *et al.*, 2017). A arquitetura *Unet* foi o primeiro modelo proposto para segmentação, este modelo apresenta a mesma arquitetura de uma rede convolucional típica. No entanto, ela não possui as camadas totalmente conectadas. A *Unet* possui um grande número de canais com

características extraídas na etapa de *Upsampling* (Figura 20), isso permite que a rede propague mais informações sobre o contexto da imagem para as camadas superiores.

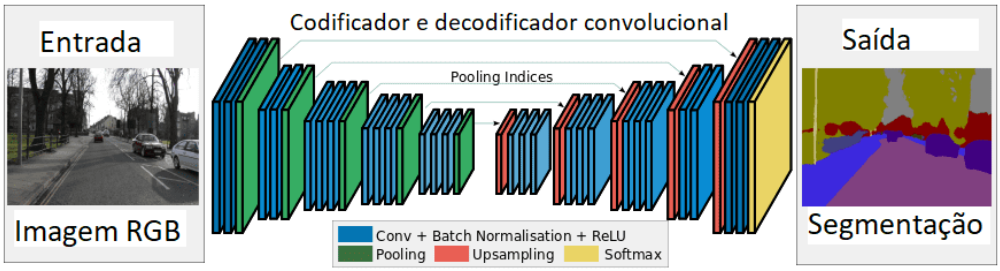
Figura 48 – Arquitetura *Unet*.



Fonte – Adaptado de (RONNEBERGER *et al.*, 2015).

A arquitetura *SegNet* é muito semelhante ao *Unet*, é composta por camadas codificadoras e decodificadoras (Figura 49), cada codificador aplica convolução, normalização por lote e uma não linearidade, em seguida, aplica o agrupamento máximo no resultado, enquanto armazena o índice do valor extraído de cada janela. Os decodificadores são semelhantes aos codificadores, a diferença é que eles não possuem uma não linearidade e fazem uma amostragem maior de sua entrada, utilizando índices armazenados no estágio de codificação. A saída é enviada para um classificador *Softmax* que fornece a previsão final.

Figura 49 – Arquitetura *SegNet*.



Fonte – Adaptado de (BADRINARAYANAN *et al.*, 2017).

Para este projeto, a arquitetura *Unet* foi selecionada para aplicar a segmentação

semântica graças à sua grande flexibilidade e compatibilidade com as bibliotecas empregadas neste trabalho.

2.3.7 *Transfer Learning*

É improvável treinar um *CNN* do zero, porque é difícil ter um conjunto de dados de tamanho suficiente. Por esse motivo, é comum pré-treinar uma *CNN* em um conjunto de dados muito grande, como o *ImageNet*, que contém 1,2 milhão de imagens com 1000 classes e, em seguida, usar a rede como um extrator de características para a tarefa de interesse. Esse processo é conhecido como *transfer learning* e apresenta três cenários principais (KARPATHY, 2020e):

a. *CNN* como extrator de características fixo: seleciona-se uma *CNN* pré-treinada em um conjunto de dados, como *ImageNet*, depois, é removida a última camada totalmente conectada. Os resultados dessa camada são as pontuações das 1000 classes de *ImageNet* e o restante da CNN é empregado como extrator de características fixo para o novo conjunto de dados. Por último, um classificador linear. Por exemplo, um classificador *SVM* ou *Softmax* é treinado para esse novo conjunto.

b. *Fine-tuning*: nesta fase, o classificador é treinado novamente na parte superior da *CNN* no novo conjunto de dados e os pesos da rede pré-treinada são ajustados com o algoritmo de retropropagação. É possível ajustar todas as camadas da *CNN* ou manter algumas das camadas anteriores fixas e ajustar apenas uma parte da rede de nível superior. Mantendo as camadas anteriores fixas, é possível preservar as camadas que executam tarefas genéricas, como detecção de bordas, detecção de manchas de cores, entre outras atividades que podem ser úteis para qualquer aplicativo. No entanto, as camadas de nível superior precisam ser ajustadas, pois são mais específicas aos detalhes das classes no conjunto de dados original.

c. Modelos pré-treinados: como o treinamento de uma *CNN* em ImageNet pode demorar entre 2 e 3 semanas, é comum encontrar bibliotecas como o *Model Zoo de Caffe* (LAPUSCHKIN, 2019), onde as pessoas compartilham seus pesos para uso público.

Quando o conjunto de dados de destino é significativamente menor que o conjunto de dados de base, o *transfer learning* pode ser uma ferramenta poderosa para permitir o treinamento de uma grande rede sem *overfit* (YOSINSKI *et al.*, 2014). No presente projeto a biblioteca *Fastai* foi utilizada para realizar o treinamento dos modelos, os detalhes gerais desta biblioteca são apresentados no apêndice A.

2.4 *MÉTRICAS DE AVALIAÇÃO DE DESEMPENHO*

Para avaliar o desempenho dos algoritmos utilizados neste trabalho e validar o treinamento, foram escolhidas as métricas mais utilizadas em trabalhos de aprendizado de máquina e segmentação semântica. Essas métricas são: *Acurácia, Precisão, Recall, F1-score* e *Intersection over Union (IoU)*, além disso foi considerado o tempo de execução de cada algoritmo. Essas métricas são descritas com mais detalhes nas subseções a seguir.

2.4.1 Acurácia

É a medida de desempenho mais intuitiva e é basicamente a razão entre as observações preditas corretamente e as observações totais (JOSHI, 2016).

2.4.2 Precisão

A precisão denota a proporção prevista de casos positivos que são corretamente verdadeiros positivos, ou seja, entre todos os casos previstos como positivos, quantos são verdadeiros (POWERS, 2020). É representada por (7).

$$Precisão = \frac{VerdadeiroPositivo}{VerdadeiroPositivo + FalsoPositivo} \tag{7}$$

2.4.3 Recall

Mede a capacidade de classificar todas as amostras positivas (JOSHI, 2016). Recall calcula quantos positivos reais o modelo captura, rotulando-o como positivo (verdadeiro positivo). Quanto mais próximo de 1, melhor será a previsão. É representada por (8).

$$Recall = \frac{VerdadeiroPositivo}{VerdadeiroPositivo + FalsoNegativo} \tag{8}$$

2.4.4 F1-score (*Jaccard index*)

É definida como a média harmônica entre Precisão e *Recall*. Essa métrica é uma medida geral da precisão de um modelo, combinando precisão e *Recall* usando (9). Ter um resultado próximo a 1 em F1 significa que se tem poucos falsos positivos e poucos falsos negativos. Portanto, identifica corretamente os resultados reais e não é afetado por resultados falsos (JOSHI, 2016).

$$F1 = 2x\frac{Precisão * Recall}{Precisão + Recall} \tag{9}$$

2.4.5 IoU

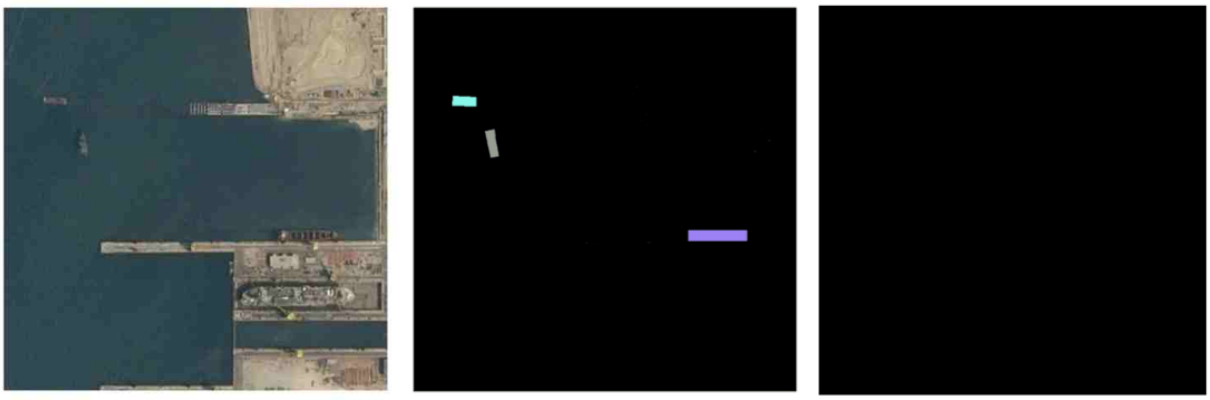
IoU, também conhecido como índice de *Jaccard*, é uma das métricas mais comuns usadas para comparar a similaridade entre formas arbitrárias. Esta métrica mede a similaridade e diversidade entre conjuntos de amostras finitas a partir da sua intersecção e união, tal como se vê em (10).

$$IoU = \frac{alvo \cap predição}{alvo \cup predição} \tag{10}$$

IoU foi a métrica implementada para medir o desempenho dos modelos na segmentação semântica. Neste caso, a acurácia não foi usada, pois pode fornecer resultados inúteis no processo de análise. Para apoiar esta afirmação é proposto o cenário mostrado na Figura 50, a imagem 50b corresponde à segmentação correta da imagem 50a, neste caso o modelo está tentando segmentar navios em uma imagem de satélite. O modelo obteve a imagem 50c como saída, supondo que haja um total de 100 pixeis na imagem e 90 deles associados ao fundo, resulta em uma acurácia de 90 % ($\frac{90}{100}$) como resposta, que aparentemente é bom, porém, neste caso não, o modelo não está conseguindo segmentar corretamente a imagem. Por outro lado, com a métrica IoU é obtido 45 % (barcos $= \frac{0}{(0+10)-0} = 0$, fundo $= \frac{90}{(100+90-90} = 0,9$, média dos dois resultados $= \frac{(0+0,9)}{2} = 0,45$), o que é mais preciso com a realidade da previsão resolvendo o problema do desequilíbrio de classes.

Figura 50 – Exemplo de Segmentação de uma imagem.

(a) Imagem de entrada (b) Segmentação correta (c) Segmentação prevista



Fonte – Adaptado de (TIU, 2019).

3 TRABALHOS RELACIONADOS

Este capítulo apresenta alguns dos principais trabalhos relacionados com o desenvolvimento de soluções tecnológicas para a detecção e classificação de defeitos em ovos comercias utilizando visão computacional.

Para determinar o estado da arte do presente projeto, foi definida a seguinte questão de pesquisa: Que soluções foram propostas para a detecção e/ou classificação de defeitos em ovos comerciais, utilizando a visão computacional?. Quando executada em bibliotecas digitais como: *IEEEXplore, Science Direct, ACM Digital Library, Springer Link*, entre outros, resultou em 66 artigos que, quando analisados sob os critérios de inclusão e exclusão, renderam 38 artigos para leitura completa. Um breve resumo dos resultados encontrados na literatura é feito abaixo.

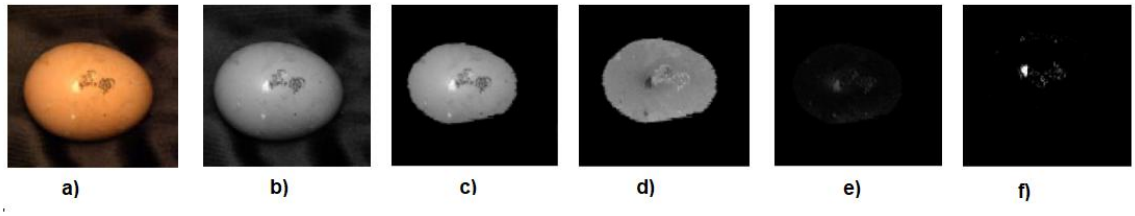
Embora as etapas de processamento dos ovos, como a coleta e embalagem, tenham sido automatizadas, ainda é necessário continuar estudando a detecção de defeitos nos ovos na etapa de classificação. Os três principais defeitos de qualidade incluem fissuras na casca, manchas internas de sangue e manchas externas de sujeira. Em relação à detecção de fissuras, está comprovado que, em circunstâncias comerciais, sensores mecânicos rápidos e não destrutivos podem ser utilizados para esse fim (MERTENS *et al.*, 2011), bem como o uso de técnicas acústicas (DE KETELAERE *et al.*, 2000; JIN *et al.*, 2015). Para a detecção de manchas de sangue interno, são utilizados métodos espectrais em que o valor sanguíneo pode ser calculado de acordo com o método descrito por (GIELEN *et al.*, 1979), usando o comprimento de onda específico para a hemoglobina (577 nm) e um comprimento de onda de referência (610 nm) (MERTENS *et al.*, 2011). Para detecção de sujeira, geralmente são usadas várias câmeras digitais (GOODRUM; ELSTER, 1992) que, usando iluminação uniforme e um sistema de processamento de imagem digital adequado, podem detectar não apenas sujeira nos ovos, mas também discriminar entre diferentes fontes de sujeira como fezes, gema, sangue etc.

Em particular, o uso de técnicas ópticas como os sistemas de visão artificial tem mostrado um crescimento exponencial neste campo, graças aos avanços no processamento de imagens e à adaptação de métodos baseados em aprendizado de máquina. Estes sistemas integram tecnologias na área da eletromecânica (hardware), instrumentação óptica (iluminação e aquisição de imagem) e processamento de imagem por computador (software). A sua utilização trouxe vantagens como a alta velocidade de resposta e precisão, proporcionando análises com alta flexibilidade e repetibilidade principalmente na indústria.

Até agora, uma grande variedade de algoritmos tem sido desenvolvida para classificação de ovos e detecção de defeitos. No ano 2000 (GARCÍA-ALEGRE *et al.*, 2000), descobriram que subtraindo a estimativa de fundo obtida por meio da filtragem

média dos formatos [(R - B) / (R + B)] a imagem diferencial de cor padronizada pode ajudar a melhorar as manchas de sujeira na casca do ovo (Figura 51). Este trabalho obteve uma precisão media na classificação de 87 % com um tempo de processamento aproximado de 100 ms.

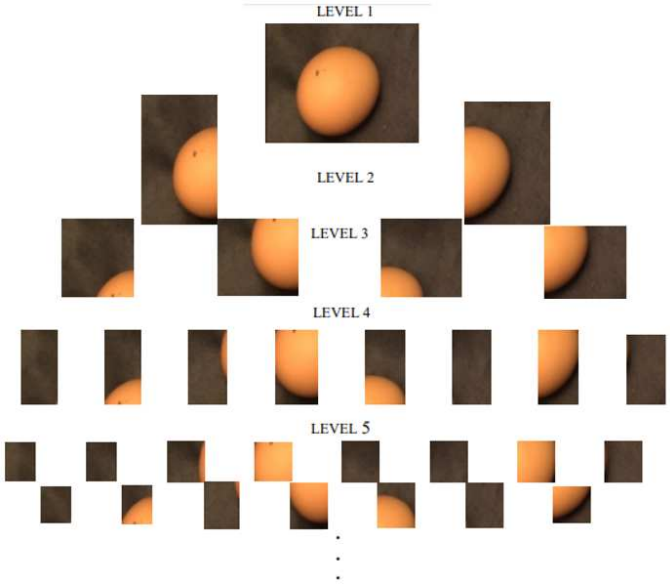
Figura 51 – (a) Imagem original, (b) Plano de cor verde, (c) Região de interesse da casca de ovo, (d) Imagem [(R-B) / (R + B)], (e) Subtração da casca de ovo para d) e (f) Imagem binária da casca de ovo.



Fonte – (GARCÍA-ALEGRE *et al.*, 2000).

Para melhorar os resultados do trabalho anterior, foi realizada uma nova investigação (RIBEIRO *et al.*, 2000), na qual foi proposto um algoritmo de classificação baseado em Algoritmos Genéticos (AG). O algoritmo usou um processo de divisão de imagens para calcular iterativamente o número de pixels defeituosos usando algoritmos evolutivos (Figura 52). Os antecedentes das regras de classificação foram obtidos por meio de um método de aprendizado baseado em exemplos. Este trabalho obteve melhores resultados na classificação em relação ao trabalho anterior.

Figura 52 – Processo de decomposição de imagem.



Fonte – (RIBEIRO *et al.*, 2000).

Em (NAKANO *et al.*, 2001), não apenas procuraram detectar manchas de sujeira,

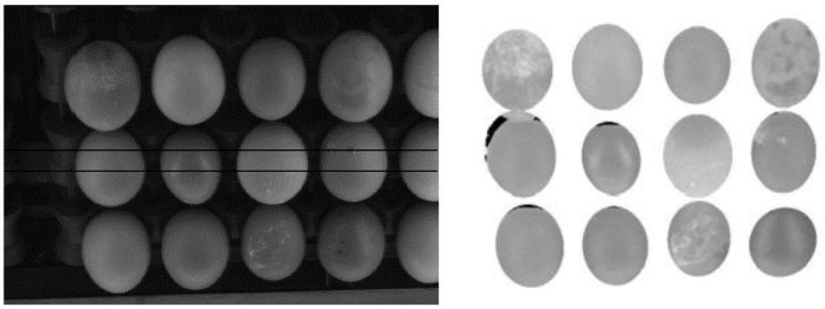
mas também detectaram outros defeitos como ovos fissurados, ovos quebrados e ovos com sangue interno. Técnicas simples de processamento de imagens foram utilizadas nos algoritmos propostos, tais como: detecção de bordas, eliminação de componentes de ruído, entre outros. Este foi um dos estudos que funcionou em tempo real mais completo e com as maiores percentagens de sucesso obtidas: 97,3 % na detecção de sujeira, 96,8 % na detecção de fissuras, 98,5 % na detecção de ovos quebrados e 95,8 % na detecção de sangue interno. No entanto, o algoritmo obteve bons resultados em ovos com casca branca, seu desempenho diminuiu para ovos com casca marrom.

Em (FEYAERTS *et al.*, 2001) tentaram detectar manchas de sujeira nos ovos marrons, diretamente na linha de produção, usando uma aproximação do contorno dos ovos (*elipse tted*) (Figura 53), mas esse algoritmo não obteve bons resultados, especialmente devido ao alto número de falsas rejeições (ver Tabela 2).

Figura 53 – Aproximação dos contornos dos ovos por elipses.

(a) Imagem de origem

(b) Área de interesse após ajuste do contorno da elipse



Fonte – (FEYAERTS *et al.*, 2001).

Tabela 2 – Resultados classificação.

Estado Real	Aceitar	Programado	Rejeitar
Aceitar	84,0	3,8	12,2
Programado	42,5	30,0	27,5
Rejeitar	34,5	7,4	58,1

Fonte – (FEYAERTS *et al.*, 2001).

Quatro anos depois, (MERTENS, Kristof *et al.*, 2005) projetaram um sistema de detecção fora de linha usando visão computacional para identificar diferentes tipos de sujeira na casca de ovo marrom, incluindo manchas escuras (fezes), manchas brancas (ácido úrico), sangue e gema. Os algoritmos propostos usaram técnicas clássicas de processamento de imagens, como o uso de histogramas para realizar a limiarização e a otimização do contraste e brilho das imagens, um filtro de partículas também foi aplicado para descartar pequenas manchas causadas por pigmentação ou pequenas

irregularidades na casca do ovo. A classificação foi baseada em operadores lógicos padrão como mostrado no Quadro 4. O sistema de visão projetado mostrou uma precisão de 99 % para a detecção de manchas de sujeira, a maior precisão alcançada em aplicativos fora de linha.

Quadro 4 – Algoritmos propostos.

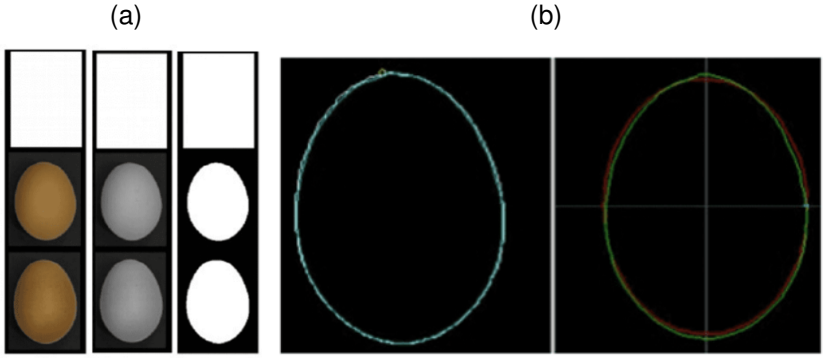
Etapa de processamento	**Manchas brancas**	**Manchas escuras**	**Manchas de sangue**	**Manchas de gema**
Imagem colorida				
Pré-processando			Operador lógico: (imagem original) XOR (cor vermelha) Cor da subtração: verde	
Extração de plano de cor	Plano azul	Plano vermelho	Plano vermelho	Plano de saturação
Imagem em escala de cinza				
Otimizando valores de brilho e contraste	Equalização do histograma do intervalo de 31 a 248 Correção gama: 2,0	Equalização do histograma do intervalo de 0 a 185	Equalização do histograma do intervalo de 77 a 181	Equalização do histograma do intervalo de 27 a 210
Limiar	150	145	150	80
Imagem binária				
Máscara de imagem **Filtro de partícula** **Análise de partículas**	Máscara de imagem 0 a 5 pixels Contagem de pixels	Máscara de imagem 0 a 5 pixels Contagem de pixels	Máscara de imagem 0 a 5 pixels Contagem de pixels	Máscara de imagem 0 a 5 pixels Contagem de pixels

Fonte – (MERTENS, Kristof *et al.*, 2005).

Até aqui, os estudos encontrados tinham como objetivo detectar sujeira, fissuras ou sangue interno; em 2008, foi encontrado um estudo para avaliar a forma do ovo (HAVLÍČEK *et al.*, 2008). Esta é uma característica indispensável para classificar os ovos por sua geometria. Para encontrar a forma do ovo, este estudo usou duas abordagens: na primeira abordagem, alguns parâmetros geométricos do ovo foram medidos manualmente, como o comprimento do ovo (L) e sua largura máxima (W). Além disso, outros parâmetros como índice de forma, área superficial e volume de ovos foram considerados. Como a medição direta precisa desses últimos parâmetros representava um problema bastante difícil, o estudo propôs diferentes fórmulas com diferentes coeficientes e valores para encontrar esses parâmetros. A segunda abordagem foi a medição quantitativa da forma através da pontuação dos componentes principais dos *Elliptic Fourier descriptors (EFDs)* empregando análise de imagem. As imagens foram submetidas a um processamento simples, como a conversão em escala de cinza e sua respectiva binarização. Os descritores de Fourier foram obtidos a partir da imagem binária (Figura 54). O estudo afirmou que o uso dos cinco primeiros componentes principais fornece um bom resumo dos dados e representa 100 % da variação total. Onde foi verificado que mais de 99 % das variações na forma dos ovos podem ser explicadas pelos quatro primeiros componentes e mais de 87 % da variação total da forma pode ser contabilizada na proporção comprimento/largura. Foi verificado pelas

duas abordagens que é possível uma descrição simples, mas satisfatória, da forma do ovo.

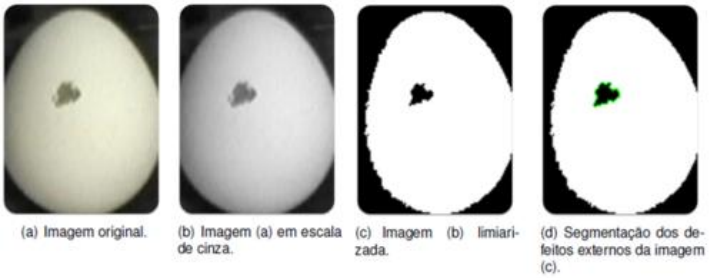
Figura 54 – a) Imagem original, imagem em escala de cinza e imagem binária. O retângulo branco situado na parte superior da imagem representa uma escala de medida, b) Contorno do ovo a partir dos descritores de Fourier.



Fonte – (HAVLÍČEK *et al.*, 2008).

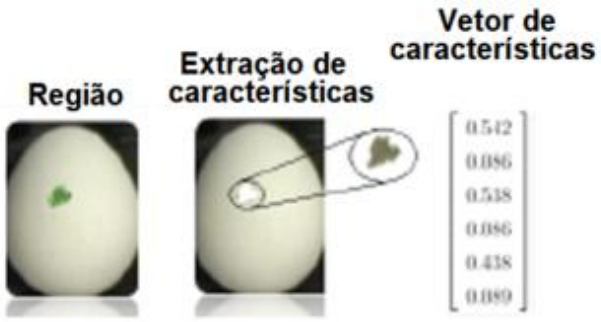
Em (MACHADO *et al.*, 2009) desenvolveram uma série de algoritmos para a detecção, o monitoramento, a contagem e a classificação de ovos, sob quatro defeitos externos: sujeira, fissuras, manchas de sangue e vazamentos de gema, e um defeito interno: pontos de sangue. Os algoritmos propostos usaram técnicas clássicas de processamento de imagens. Para realizar a segmentação, eles usaram um método conhecido como *Subtração de Fundo Adaptativo*, que identifica as partes da imagem que estão em movimento, como no caso dos ovos. Uma vez identificados os ovos, os algoritmos de detecção de defeitos são executados. Para detectar defeitos externos, tais como sujidade, foi utilizado o método de componentes conexos para identificar a região do defeito (Figura 55). Desta região, foi extraído um vector característico, constituído pelas médias estatísticas e variações estatísticas das cores RGB (Figura 56). Este vector foi introduzido numa rede neural artificial *MLP* para efetuar a classificação.

Figura 55 – Resultado da etapa de segmentação dos defeitos externos.



Fonte – (MACHADO *et al.*, 2009).

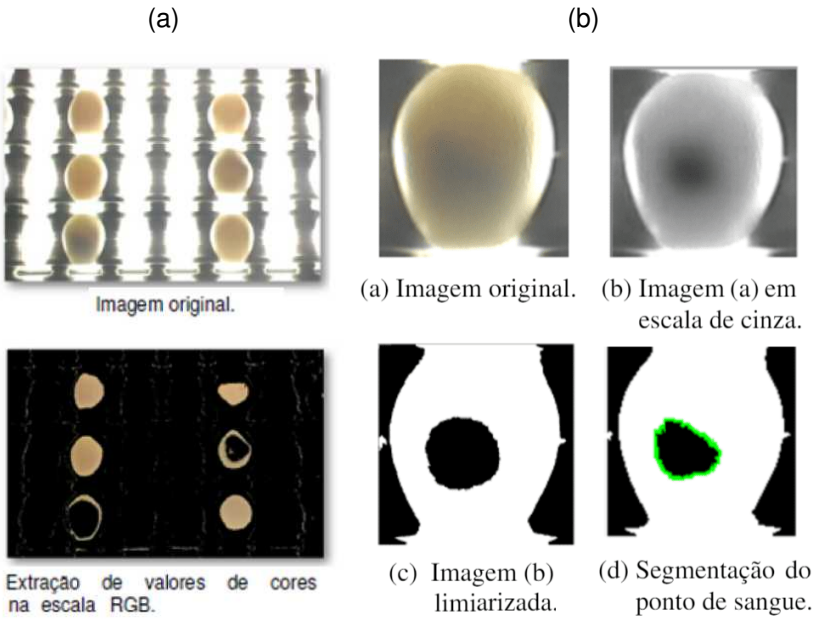
Figura 56 – Extração de características.



Fonte – (MACHADO *et al.*, 2009).

Para a detecção de fissuras, foi realizada uma análise de detecção de bordas, em que cada borda foi comparada com um limiar para definir a presença ou ausência de fissuras. Por último, para detectar defeitos internos como manchas de sangue, a segmentação foi feita com base na extração de uma gama de valores de cor na escala RGB com valores de 170 a 220 para o canal vermelho, 80 a 180 para o canal verde e 80 a 160 para o canal azul (Figura 57a). O processo seguinte foi o mesmo que o utilizado para a detecção de sujidade (Figura 57b). A abordagem proposta foi avaliada obtendo taxas de precisão de 100 % na contagem de ovos, 75,6 % em detecção de sujeira, 73,3 % na detecção de trincas, 78,5 % na detecção de manchas de sangue, 62,5 % na detecção de vazamentos de gemas e 50 % na detecção de pontos de sangue para ovos brancos.

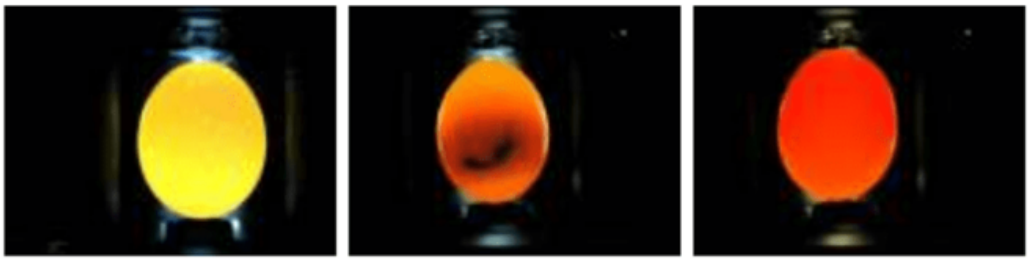
Figura 57 – a) Segmentação das manchas de sangue interno, b) Resultado da etapa de segmentação dos defeitos internos.



Fonte – (MACHADO *et al.*, 2009).

Em (DEHROUYEH *et al.*, 2010), foi proposto um algoritmo baseado no processamento de imagens para detectar manchas de sangue interno e sujeira da casca de ovo sob diferentes iluminações. Eles foram um dos primeiros a usar o espaço de cores *HSI*, especialmente o histograma relacionado ao matiz (*H*) para a detecção de manchas de sangue. A partir de uma série de testes, eles descobriram que em ovos intactos, o matiz tinha mais frequência entre 40 e 60, nos ovos com mancha de sangue concentrada, o histograma tinha um matiz na faixa de 200 a 359, porque a mancha parecia escura e próxima da cor preta. Finalmente, nos ovos onde o sangue se espalhou dentro do ovo como mostrado na Figura 58c, o componente do matiz tinha mais frequência entre 0 e 20. Por outro lado, para a detecção de sujeira, foi empregado o método de detecção de bordas *Canny* e depois, uma análise de componentes conexos para a tomada de decisões foi feito. Este trabalho obtive uma precisão media na diferenciação de manchas de sangue interno de 91 % e de 85,66 % para manchas de sujeira.

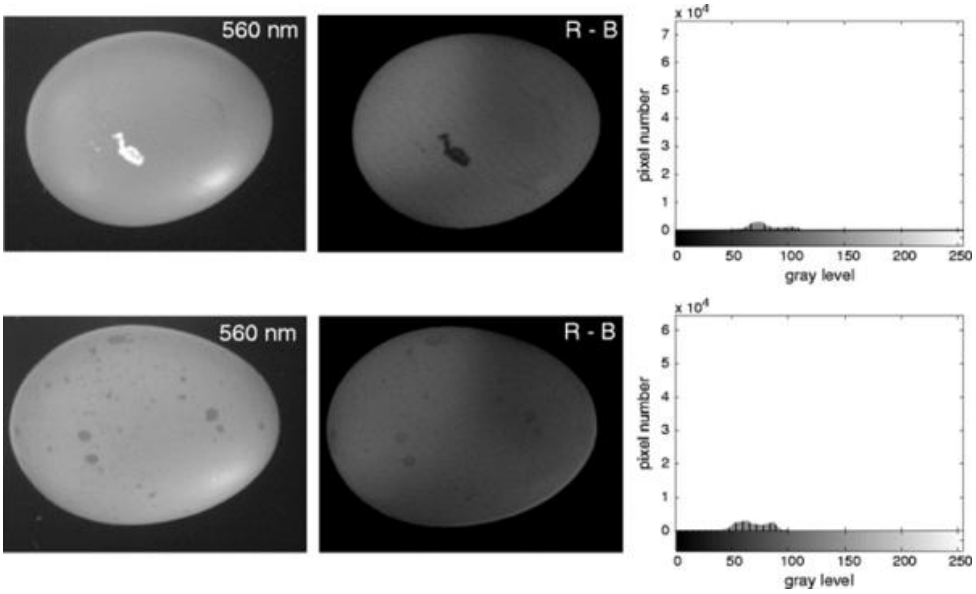
Figura 58 – (a) Uma amostra intacta, (b) e (c) amostras defeituosas.



Fonte – (DEHROUYEH *et al.*, 2010).

Em (LUNADEI *et al.*, 2012) desenvolveram um aplicativo fora de linha que combinava oportunamente imagens monocromáticas para detectar trincas e manchas de sujeira nos ovos. Este estudo propôs que, para obter uma imagem com alta discrepância entre os valores de pixel do fundo do ovo e os pontos de sujeira, era necessário extrair o canal azul (B, 430- 490 nm) do canal vermelho (R, 620-780 nm) (R menos B (R - B)) como mostrado na Figura 59. O algoritmo para a detecção de sujeira foi baseado em técnicas simples, como o método *Otsu*, para realizar a binarização das imagens, operações morfológicas para eliminar possíveis ruídos e intensificar as possíveis manchas e métodos de rotulagem para fazer a classificação. O algoritmo conseguiu classificar corretamente quase 98 % das amostras com um tempo de processamento de aproximadamente 50 ms, sendo o menor tempo possível relatado em aplicativos fora de linha.

Figura 59 – Imagens digitais adquiridas a 560 nm (à esquerda), imagens R - B (no centro) e histogramas de imagens R - B de uma casca de ovo com fezes brancas (painel superior) e com manchas escuras naturais (painel inferior).

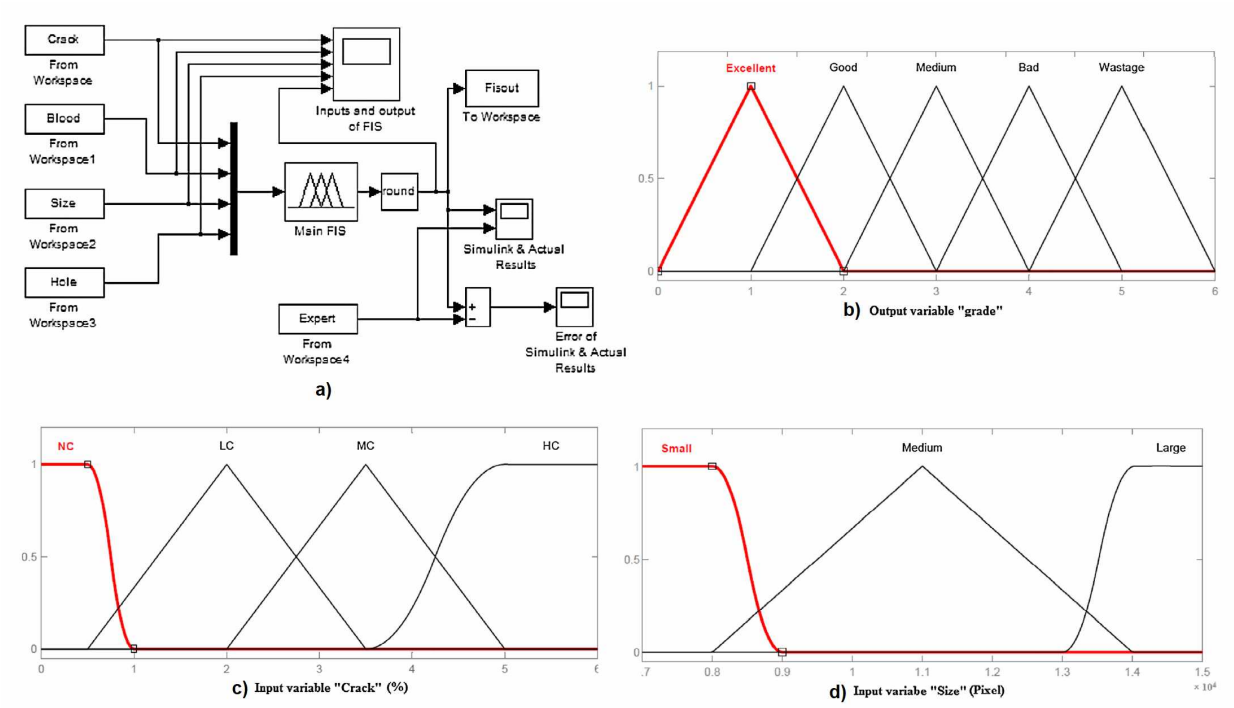


Fonte – (LUNADEI *et al.*, 2012).

Em (OMID *et al.*, 2013), foi desenvolvido um sistema inteligente baseado em *lógica difusa* e técnicas de visão artificial para a classificação dos ovos por tamanho e tipo de defeito. Os defeitos detectados foram manchas de sangue internas, fissuras e quebras na casca do ovo. O espaço de cor *HSV* foi utilizado para obter características visuais durante a fase de segmentação da imagem. Técnicas simples no processamento de imagens, tais como operações morfológicas e contagem de píxeis foram utilizadas para determinar o tamanho dos ovos. Para identificar fissuras foi utilizado o método de detecção de bordas *LoG* e para o reconhecimento da mancha de sangue foram baseados no método proposto em (DEHROUYEH *et al.*, 2010). Como método de classificação um *Fuzzy Inference System (FIS)* foi utilizado (Figura 60a). O sistema classificou o ovo em cinco classes, excelente (A), bom (B), médio (C), ruim (D) e desperdício (E) (Figura 60b). Quatro variáveis (fissura, mancha de sangue, tamanho e quebra) foram definidas como entradas para o sistema Fuzzy.

Para a fissura da casca do ovo, foram definidos quatro conjuntos difusos, sem fissura (NC), rachadura baixa (LC), rachadura média (MC) e rachadura alta (HC) (Figura 60c) e para o tamanho, foram especificadas três séries: pequenas, médias e grandes (Figura 60d). A precisão obtida foi de 95 % para a detecção de tamanho, 94,5 % para a detecção de fissuras y 98 % para a detecção de quebra. Quanto à detecção de sangue interno, basearam-se apenas na precisão relatada em (DEHROUYEH *et al.*, 2010), que foi de 91 %. A precisão geral do modelo *FIS* para a classificação de ovos foi de 95,4 %.

Figura 60 – O modelo *FIS* desenvolvido.



Fonte – (OMID *et al.*, 2013).

Entre 2014 e 2017, a maioria dos artigos encontrados foram aplicativos fora de linha que não excederam a detecção de mais de um defeito como é apresentado no Quadro 5.

Quadro 5 – Trabalhos entre 2014 e 2017

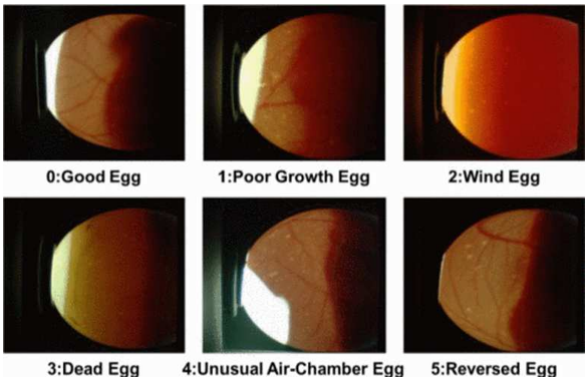
Trabalho	**Acurácia obtida na detecção de defeitos e características do ovo**					**Técnica empregada**
	Fissuras	**Quebra**	**Sujeira**	**Sangue interno**	**Tamanho**	
(WARANUSAST *et al.*, 2016) .	Não avaliado	Não avaliado	Não avaliado	Não avaliado	80,4 %	Técnicas clássicas de processamento de imagem
(ZALHAN *et al.*, 2016)	Não avaliado	Não avaliado	Não avaliado	Não avaliado	96 %.	Técnicas clássicas de processamento de imagem
(PRIYADUMKOL *et al.*, 2017),	94 %	Não avaliado	Não avaliado	Não avaliado	Não avaliado	Técnicas clássicas de processamento de imagem
(ABDULLAH *et al.*, 2017)	90.6 %	Não avaliado	Não avaliado	Não avaliado	Não avaliado	Técnicas clássicas de processamento de imagem e *Support Vector Machine (SVM)* para classificação.

Fonte – O autor.

Até este momento, a maioria dos trabalhos usava técnicas clássicas de processamento de imagens, nas quais eram estabelecidos limites fixos para a tomada de

decisões, por exemplo, se a área da mancha for maior que *x*, o ovo é considerado sujo. Outros trabalhos utilizaram métodos ligeiramente mais avançados para realizar a classificação, como treinamento de redes neurais, sistemas difusos, classificadores *SVM*, entre outros. Uma exceção notável foi apresentada em (SHIMIZU *et al.*, 2017) que não era apenas destinada à detecção de vários defeitos nos ovos, mas também foi o primeiro e único artigo encontrado que utilizava técnicas de *aprendizagem profundo* para esse fim. Este estudo aplicou uma *CNN* para a inspeção de qualidade dos ovos, discriminada em seis categorias: ovo bom, ovo em crescimento fraco, ovo sem gema, ovo morto, ovo com câmara de ar incomum e ovo invertido (Figura 61).

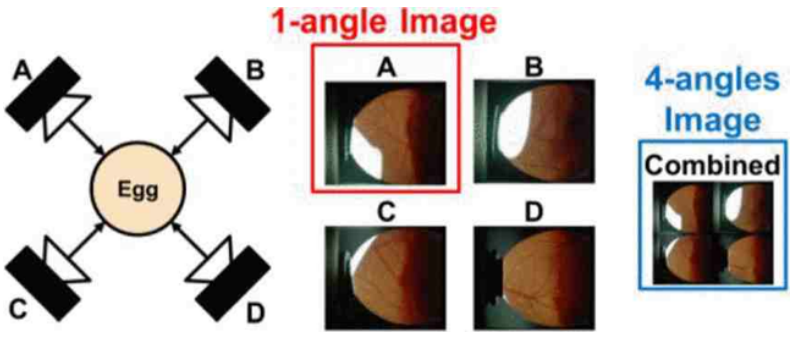
Figura 61 – Um exemplo de conjunto de dados de ovo



Fonte – (SHIMIZU *et al.*, 2017).

Procurou-se não apenas classificar defeitos nos ovos, mas também determinar entre dois conjuntos de dados, qual dos dois conjuntos era melhor para alimentar a rede. O primeiro conjunto consistiu em imagens dos ovos capturados em um único ângulo e o segundo conjunto forneceu 4 ângulos diferentes para cada ovo, conforme mostrado na Figura 62. Como resultado, foi obtida uma precisão total na classificação para o primeiro conjunto de 89 % e para o segundo conjunto de 92,3 %.

Figura 62 – Fotografia do ovo e imagens de entrada.

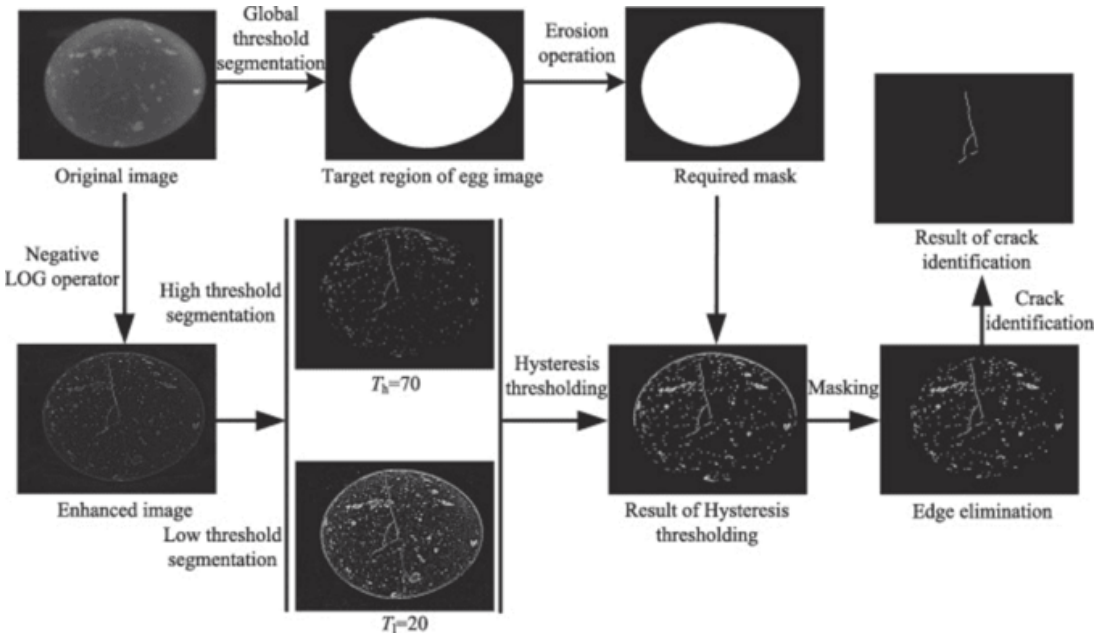


Fonte – (SHIMIZU *et al.*, 2017).

No ano 2019, foram encontrados dois artigos para aplicação em tempo real,

um deles foi (GUANJUN *et al.*, 2019) onde se concentraram em determinar apenas rachaduras na casca do ovo. Para detectar rachaduras, eles usaram uma série de métodos (Figura 64), começando com o operador negativo *LoG* para suprimir o ruído na imagem. Junto com esse método, o operador *Laplace* foi usado para obter a imagem binária. Como método de segmentação, foi utilizado o limiar de histerese com dois limiares (limiar alto = 70 e limiar baixo = 20). Finalmente, o índice melhorado *Local Fitting Image (LFI)* foi usado para identificar fissuras. Esse índice foi composto por um total de 100 rachaduras de diferentes formas e direções, além de 100 regiões sem rachaduras. Como resultado, eles obtiveram uma taxa de sucesso na detecção de fissuras de 92,5 %.

Figura 63 – Fluxograma do algoritmo de identificação de trinca de ovo baseado em visão artificial.



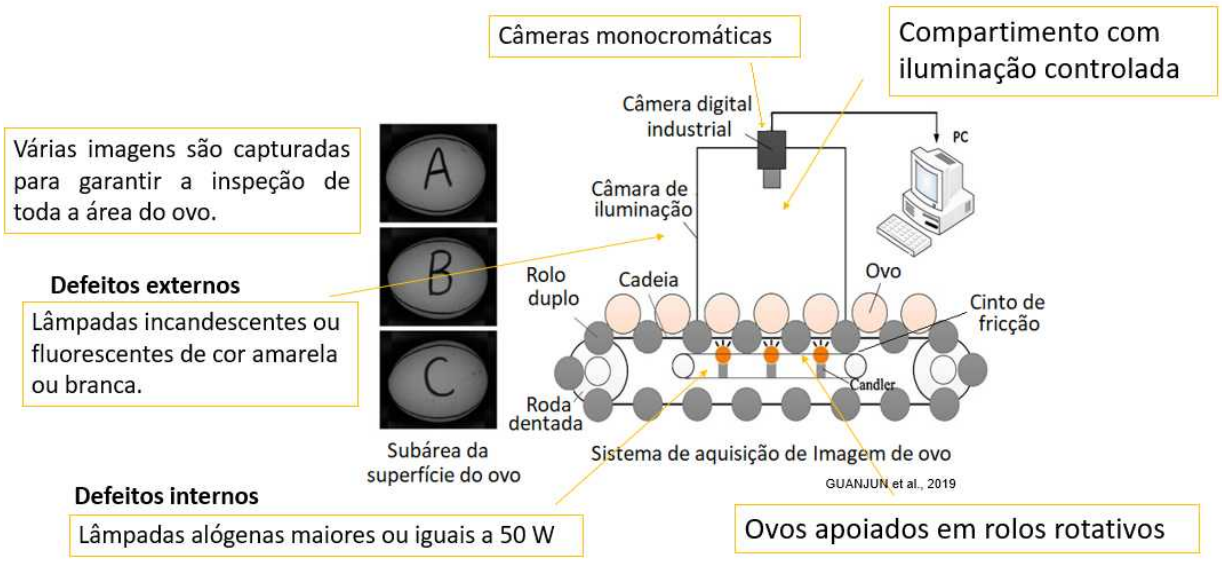
Fonte – (GUANJUN *et al.*, 2019).

Outro dos artigos foi (ALON, 2019) onde propuseram uma série de algoritmos para detectar várias rachaduras, sujeira e defeitos nos ovos. Eles usaram técnicas simples de processamento de imagem, como transformações de formato (*RGB* para *HSV* para detecção de sujeira e *RGB* para *YIQ* para detecção de trincas), normalização de imagem, filtragem e binarização. Após os testes, o sistema obteve taxas de sucesso entre [90 e 93,3] % na determinação da presença de rachaduras ou sujeira nos ovos.

Como consideração final, é necessário destacar os equipamentos utilizados na maioria dos processos. Para a captura das imagens foi empregado um compartimento com iluminação controlada, onde os ovos eram apoiados em rolos rotativos. À medida que os ovos se movem ao longo das linhas de transporte, várias imagens são capturadas para garantir a inspeção de toda a área do ovo. Para a detecção de defeitos

internos, como manchas de sangue, ausência de gema ou rachaduras na casca do ovo, foram utilizadas lâmpadas alógenas maiores ou iguais a 50 W. Para destacar defeitos externos, como sujeira, foram utilizadas lâmpadas incandescentes ou fluorescentes de cor amarela ou branca. Essas lâmpadas estavam localizadas na parte superior ou nas laterais do compartimento. Finalmente, câmeras monocromáticas ou coloridas *CCD* foram usadas para capturar as imagens.

Figura 64 – Equipamentos geralmente utilizados nos processos.



Fonte – Adaptado de (GUANJUN *et al.*, 2019).

4 DESENVOLVIMENTO DO PROJETO

Este capítulo apresenta a metodologia proposta para o desenvolvimento de um sistema automático de inspeção visual de defeitos em ovos comerciais. É apresentada uma descrição dos materiais implementados e dos algoritmos desenvolvidos para a classificação dos ovos.

A primeira etapa da metodologia foi definir o conjunto de requisitos para o desenvolvimento do sistema de inspeção visual de ovos, requisitos levantados em conjunto com a empresa *Plasson do Brasil* responsável pelo desenvolvimento de soluções em equipamentos e serviços para avicultura e suinocultura. Adicionalmente, foi realizado um mapeamento sistemático da literatura com o objetivo de adquirir informações relevantes sobre os métodos propostos e os equipamentos utilizados até o momento para detectar e/ou classificar defeitos em ovos comerciais. A lista de requisitos funcionais e não funcionais definidos para este trabalho se apresentam a seguir:

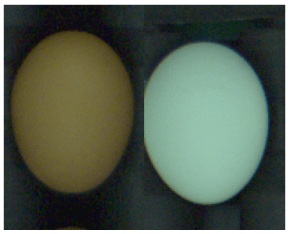
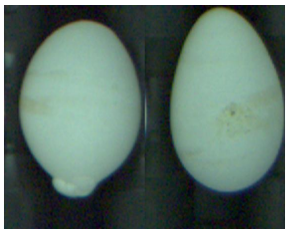
a. Requisitos funcionais:

   - O sistema de inspeção deve classificar ovos brancos e vermelhos de acordo com as quatro categorias apresentadas no Quadro 6. Os defeitos de sujeira, geometria e fissuras foram selecionados por serem os defeitos mais comuns em ovos.
   - O sistema de inspeção deve permitir definir o grau de sujeira a ser descartado, desde pequenas manchas até descartar apenas manchas maiores de sujeira nos ovos. Esse requisito foi definido porque algumas indústrias preferem apenas realizar a classificação dos ovos muito sujos e passar para a linha de embalagem os ovos com pequenas manchas que não afetam a qualidade do produto. É por isso que a opção deve ser deixada em aberto para decidir se deve descartar todos os tipos de manchas ou tamanhos específicos de manchas.
   - O sistema deve ser capaz de ser configurado para definir as geometrias de ovo aceitáveis e as geometrias de ovo que devem ser rejeitadas.

b. Requisitos não funcionais:

   - O motor associado ao sistema que permite a movimentação dos ovos deve operar a uma frequência de 60 Hz. Nessa frequência com o hardware de inspeção usado na indústria é possível processar em média 4.1 ovos por segundo. Este requisito foi definido de acordo com a operação real do sistema.
   - O sistema deve ser compatível com o módulo de transporte de ovos utilizados na indústria. Neste caso os ovos são apoiados em rolos cônicos posicionados simetricamente em uma corrente giratória.

Quadro 6 – Categorias a serem classificadas pelo sistema de inspeção.

Categoria	Observações	Imagem
Ovos normais	Ovos limpos, que podem apresentar pequenas manchas de sujeira, mas não prejudicam a aparência geral limpa do ovo. Essas manchas não devem cobrir mais do que 1/32 da superfície do ovo. Ovos com formato aproximadamente normal e que não apresentem fissuras na casca.	
Ovos sujos	Ovos com manchas de sujeira que cobre aproximadamente mais do que 1/32 da superfície do ovo são considerados levemente sujos e ovos com mais do que 1/4 de sujeira são considerados muito sujos. Divisão da superfície do ovo.  Fonte - (HAUVER, 1961).	 
Ovos com geometria anormal	Ovos que têm uma geometria diferente de uma forma circular normal.	
Ovos fissurados	Ovos com furos na casca ou grandes fendas.	

Fonte – O autor.

A partir da lista de requisitos e das informações coletadas no mapeamento sistemático, foi desenhado um protótipo para a inspeção visual dos ovos. Este protótipo é composto por dois módulos principais, um módulo associado a um compartimento fechado para garantir condições de iluminação constantes na captura das imagens e

um módulo de movimentação que permite a rotação e translação dos ovos através do compartimento por meio de uma série de rolos cônicos dispostos simetricamente sobre uma corrente rotativa. Foi criado um novo banco de imagens em formato *BGR* com duas câmeras posicionadas na parte superior do compartimento fechado, essas câmeras adquiriam imagens com resolução de 1280 x 1024 pixels que capturam entre 0 a 18 ovos cada.

Para este projeto, diversos métodos de detecção e classificação de defeitos em ovos comerciais foram propostos utilizando técnicas clássicas e técnicas baseadas em aprendizagem profundo. Os algoritmos com técnicas clássicas foram o resultado de uma hibridização e modificação de diferentes algoritmos propostos na literatura. Para o desenvolvimento das técnicas baseadas em aprendizagem profundo, optou-se por implementar dois métodos, o primeiro método foi a classificação de imagens e o segundo método foi a segmentação semântica, avaliando diferentes arquiteturas de *CNNs* com diferentes profundidades *ResNet18*, *ResNet34*, *ResNet50* e *GoogLeNet*. Essas arquiteturas foram selecionadas com base nos requisitos de velocidade e precisão de acordo com uma análise realizada em (STANFORD, 2017d). Todos os modelos de aprendizagem profundo foram pré-treinados no banco de imagens *ImageNet* e treinados no banco de dados criado para este projeto.

Os métodos propostos foram submetidos a duas fases de experimentação, a primeira fase destinou-se a avaliar o desempenho de cada modelo individualmente utilizando as métricas de avaliação apresentadas na seção 2.4. A segunda fase de experimentação consistiu na comparação de todos os métodos propostos, tanto clássicos quanto baseados em aprendizagem profunda, em um único conjunto de dados, a fim de responder às questões de pesquisa levantadas neste projeto. Esses experimentos e seus resultados são descritos com mais detalhes no Capitulo 5.

4.1 MATERIAIS

Esta seção descreve os componentes do sistema de aquisição desenvolvido para este trabalho e as informações gerais da base de imagens obtida para o desenvolvimento dos algoritmos.

4.1.1 Sistema de aquisição

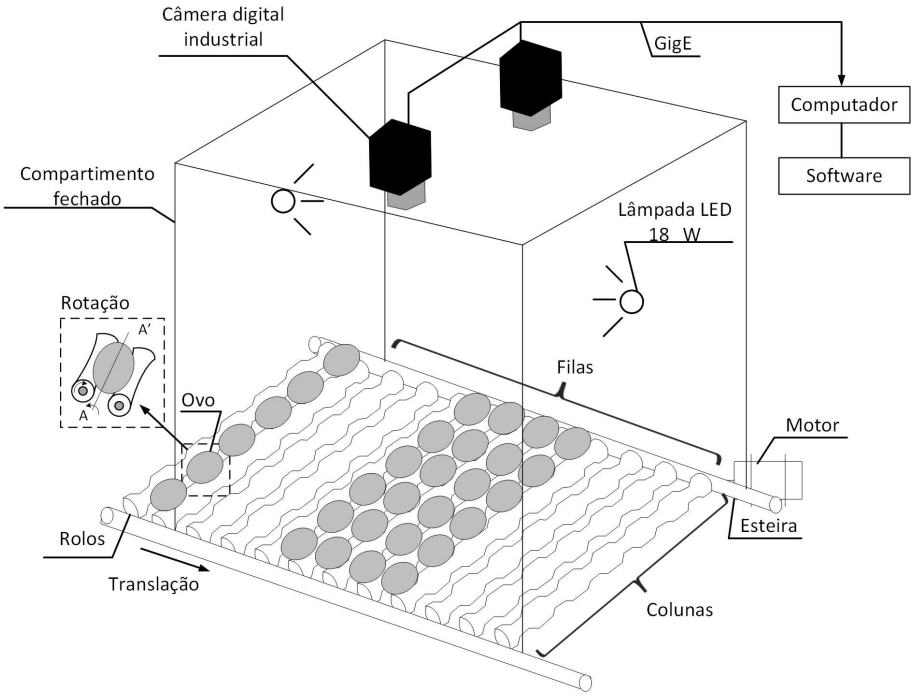
Observando os requisitos a serem atendidos, foi proposto um sistema de aquisição de imagens empregando diferentes componentes eletromecânicos. O projeto de tal sistema foi elaborado de forma a atender especificações levantadas junto à empresa parceira, de modo a ser compatível com componentes, equipamentos e condições reais de utilização na indústria.

O sistema de aquisição é composto por um compartimento fechado de (500 x

500 x 500) mm com paredes brancas, de forma a garantir uma iluminação uniforme e evitar qualquer efeito da luz ambiente. Uma iluminação direcional foi escolhida dentro do compartimento colocando duas lâmpadas tabulares em duas paredes paralelas do compartimento. Como tipo de iluminação, optou-se pela iluminação LED por fornecer luz difusa e apresentar diversas vantagens conforme mencionado na seção 2.1. Foram utilizadas lâmpadas com uma potência de 18 W com uma temperatura de cor de 6500K. Para capturar as imagens, foram selecionadas duas câmeras industriais *Basler acA1300-60 gc* com lente varifocal de 4-12 mm e zoom manual de 1/2 polegada. As câmeras foram posicionadas na parte superior do compartimento e conectadas a um computador através de uma conexão GigE para transferência de dados. Essas câmeras adquirem imagens em RGB a 60 fps com resolução de (1280 x 1024) px. Seis colunas de ovos foram colocadas em rolos cônicos de 335 mm de comprimento com um intervalo de 28 mm entre eles dispostos simetricamente em uma corrente de rolamento.

A corrente é girada com um motor trifásico que movimenta os roletes. Este movimento permite que cada ovo seja movido para frente juntamente com um movimento de rotação em torno do eixo A-A', conforme mostrado na Figura 65. As câmeras são acionadas a partir de um pulso externo gerado por um Controlador Lógico Programável (*CLP*) que permite que as capturas sejam sincronizadas em relação ao movimento da corrente de rolamento.

Figura 65 – Protótipo para inspeção de ovos usando visão computacional.

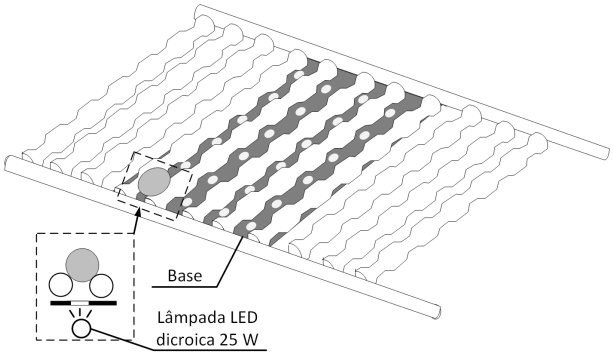


Fonte – O autor.

Os ovos também foram iluminados por baixo. Para isso, foram selecionadas

lâmpadas LED dicróicas de 25 W com temperatura de cor de 6500 K. Essas lâmpadas foram instaladas na parte inferior do compartimento conforme mostrado na Figura 66. Para evitar reflexos de luz, cada lâmpada foi posicionada abaixo de uma base com orifícios localizados em cada ovo.

Figura 66 – Iluminação inferior no protótipo para inspeção.



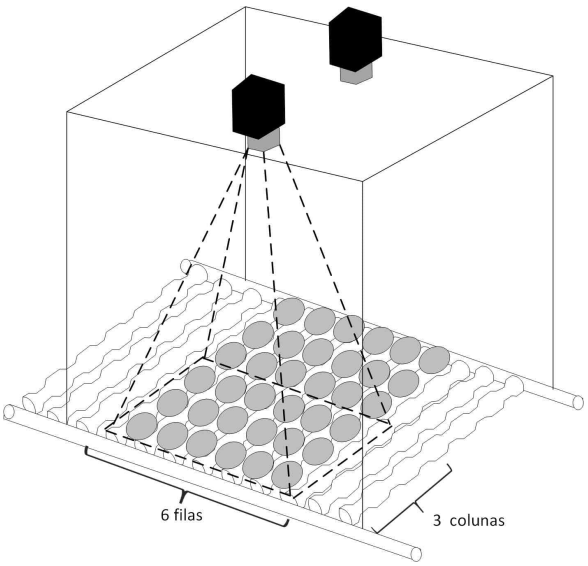
Fonte – O autor.

Maiores detalhes dos componentes e da implementação assim como imagens do protótipo construído são apresentados no apêndice B.

4.1.2 Criação da Base de Imagens

O sistema de aquisição foi utilizado na criação de uma base de imagens, utilizada ao longo do desenvolvimento dos algoritmos descritos no presente trabalho. Neste sistema cada câmera captura 6 linhas por 3 colunas de ovos conforme mostrado na Figura 67.

Figura 67 – Captura de imagens pelo sistema de inspeção desenvolvido.

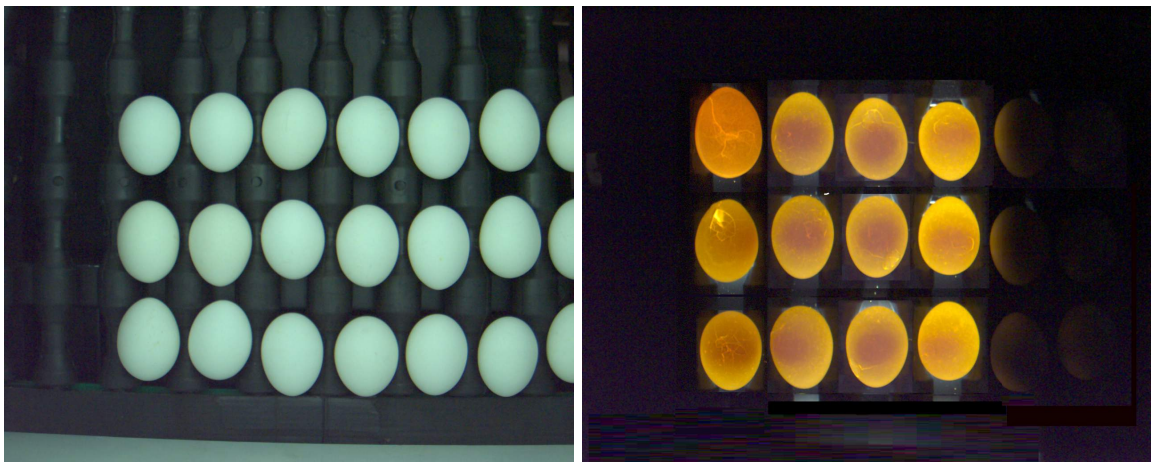


Fonte – O autor.

As imagens são adquiridas com a corrente em movimento usando um tipo de iluminação para cada captura. Um exemplo de uma imagem capturada com iluminação superior e iluminação inferior é apresentado nas Figuras 68a e 68b, respectivamente.

Figura 68 – Imagens adquiridas pelo protótipo desenvolvido utilizando (a) iluminação superior e (b) iluminação inferior.

(a) (b)



Fonte – O autor

No total, foram adquiridos 600 ovos brancos e vermelhos fornecidos pela industria, os quais foram introduzidos diversas vezes no módulo de movimentação, o que permitiu a obtenção de 46.325 imagens com as categorias apresentadas na Tabela 3.

Tabela 3 – Classificação do banco de imagens.

Tipo de iluminação utilizada	**Imagem**	**Categoria**	**Total imagens**
Iluminação superior	Ovo branco	Ovos normais	7702
		Ovos sujos	3930
		Ovos com geometria anormal	1432
	Ovo vermelho	Ovos normais	6925
		Ovos sujos	5355
		Ovos com geometria anormal	1679
	Imagem sem ovo	Vazio	17800
Iluminação inferior	Ovo branco	Ovos normais	412
		Ovos fissurados	179
	Ovo vermelho	Ovos normais	471
		Ovos fissurados	240
	Imagem sem ovo	Vazio	200

Fonte – O autor.

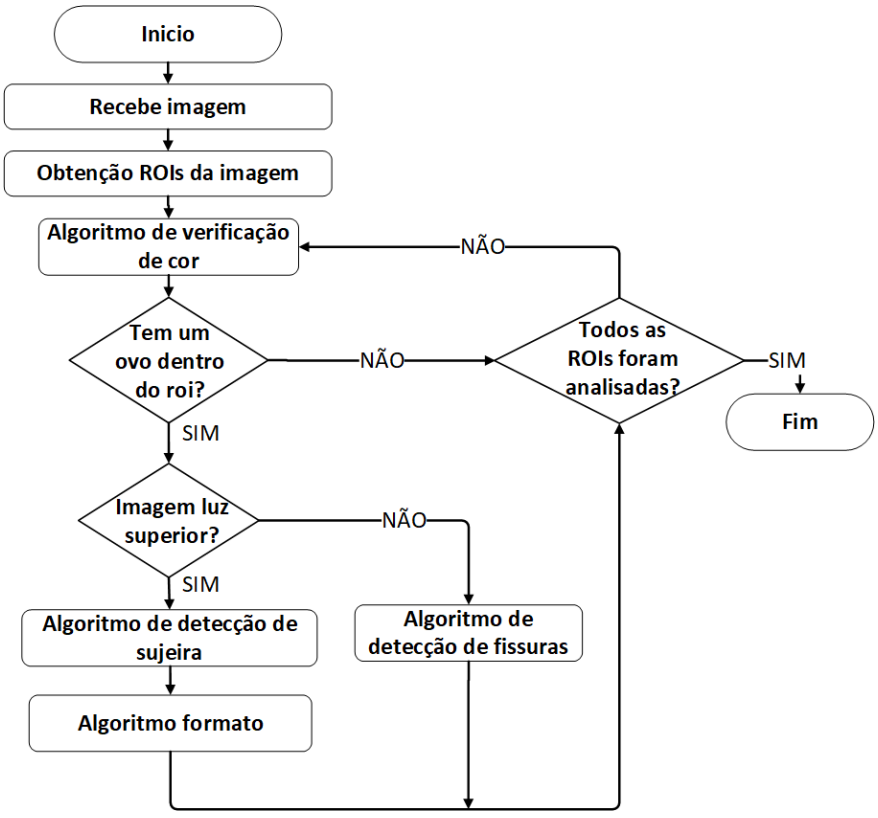
As configurações do sistema para a captura das imagens são apresentadas em mais detalhes no apêndice C.

4.2 ALGORITMOS DE CLASSIFICAÇÃO DE OVOS USANDO TÉCNICAS CLÁSSICAS DE PROCESSAMENTO

Um dos requisitos funcionais do sistema é classificar os ovos em 4 categorias: normais, sujos, geometricamente anormais e quebrados. Para isso, foi proposto um conjunto de algoritmos que utilizam técnicas clássicas de processamento de imagens que serão descritos ao longo desta seção. Esses algoritmos foram desenvolvidos em linguagem *Python* com a biblioteca *OpenCV* versão 4.5.0.

Para detectar defeitos de sujeira e geometrias anormais nos ovos, optou-se por trabalhar com imagens capturadas com iluminação superior, por apresentarem uma faixa maior de valores que permite extrair diferentes características nos ovos, como diferenciação das manchas por cores e uma adequada captura da superfície do ovo para análise de formato. Por outro lado, para detectar defeitos associados a fissuras, verificou-se que é mais fácil detectar esses defeitos em imagens capturadas com iluminação inferior (veja apêndice B), pois as fissuras permitem a passagem de luz pela casca, o que gera uma melhor diferenciação do defeito em relação ao fundo e ao ovo. Para classificar esses defeitos, foram propostas as atividades apresentadas no fluxograma da Figura 69.

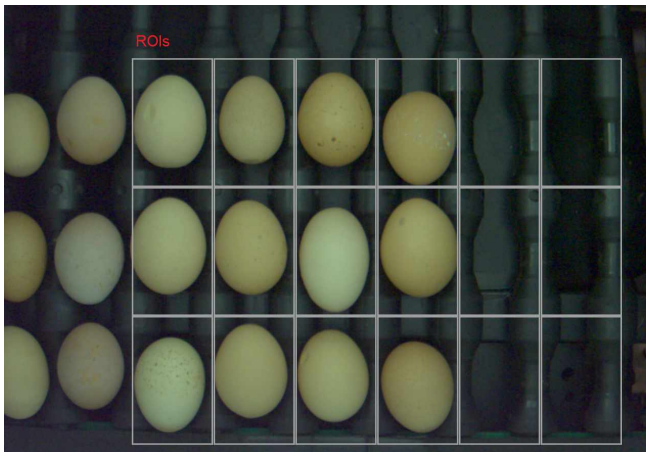
Figura 69 – Diagrama de fluxo do processo na etapa de detecção de defeitos usando iluminação superior.



Fonte – O autor.

Como primeiro passo da sequência de atividades, a imagem é recebida. Esta imagem pode conter entre 0 a 18 ovos. Como é fundamental analisar cada ovo individualmente, são definidas as regiões de interesse (*ROIs*) associadas a cada ovo. Essas regiões são quadros predefinidos organizados com base na localização dos ovos nos rolos, conforme mostrado na Figura 70. Essas regiões são constantes durante todas as imagens porque as câmeras estão sincronizadas com a corrente rotativa o que garante que no momento das aquisições os ovos apareçam sempre na mesma posição.

Figura 70 – Regiões de interesse predefinidas com base na localização dos ovos nos rolos.



Fonte – O autor.

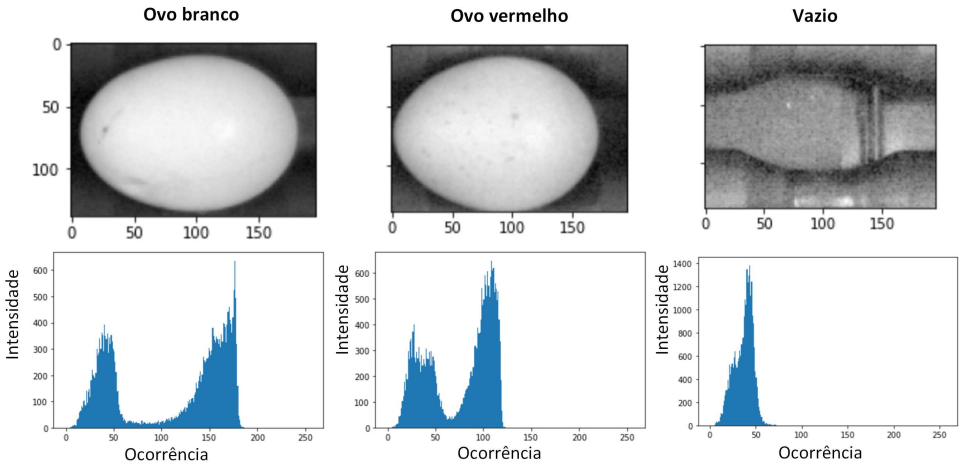
Uma vez obtidos os *ROIs*, são submetidos a um algoritmo de verificação para determinar se há ou não um ovo dentro do *ROI*, se for positivo, a cor do ovo é definida. No caso de uma imagem capturada com iluminação superior, dois algoritmos são aplicados, o algoritmo de detecção de sujeira e o algoritmo de formato. No caso de ser uma imagem capturada com iluminação inferior, é aplicado o algoritmo de detecção de fissuras. Esses quatro algoritmos serão descritos em mais detalhes nas seções seguintes.

4.2.1 Algoritmo de verificação de cor

Este algoritmo tem como objetivo verificar se existe um ovo dentro do *ROI* e determinar se a cor do ovo é branca ou vermelha. Este algoritmo usa uma operação de limiarização básica a partir de uma gama de valores de intensidade de pixel. Cada pixel da imagem é avaliado entre um limite inferior e um limite superior, os pixels entre esses dois limites são atribuídos a um valor de intensidade de 255 e os pixels fora desses limites um valor de intensidade de 0. Para determinar os limites [*inferior, superior*] a serem utilizados nesta etapa, o histograma de cada imagem no formato cinza foi analisado (Figura 71), onde pode-se observar que os ovos brancos diferem das duas

categorias vermelho e vazio por apresentarem maior intensidade nos pixels superiores a 140. Por esta razão a faixa [140, 255] foi definida para realizar a limiarização.

Figura 71 – Análise de histograma das imagens em formato cinza.

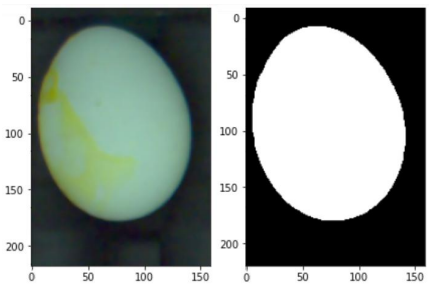


Fonte – O autor.

Como resultado da limiarização, obtém-se uma imagem em preto e branco, com os pixels brancos associados ao ovo e os pixels pretos ao fundo. Como é possível que alguns ovos vermelhos tenham uma leve coloração branca em sua superfície, é analisada a área total dos pixels brancos da imagem. A partir de uma série de experimentos verificou-se que *ROIs* com ovos brancos obtiveram áreas maiores que 13000 e ROIs com ovos vermelhos áreas menores que 10.000. Por este motivo, foi definido que ovos com áreas maiores ou iguais a 12000 são classificadas como ovos brancos e ovos com áreas menores que 12000 são classificados como possíveis ovos vermelhos.

Uma vez que um *ROI* é identificado como um ovo branco, passa por um processamento adicional para aumentar os detalhes da máscara. Para isso, a faixa do limite inferior da limiarização é diminuída em [80-255] e pequenos ruídos na imagem são eliminados com operações de filtragem. Finalmente é obtida a máscara do ovo branco mostrada na Figura 72.

Figura 72 – Máscara binária do ovo branco.

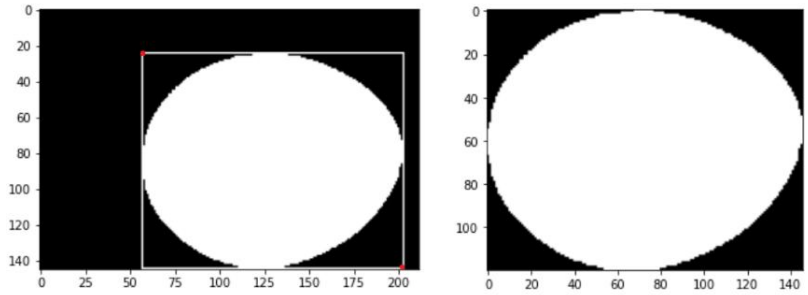


Fonte – O autor.

No caso dos *ROIs* que não são classificados como brancos, o componente R é extraído da imagem *BGR*. Uma binarização simples é aplicada com uma faixa de valores de [56-255] e pequenos ruídos são removidos com operações de filtragem. Como resultado, uma máscara binária é obtida onde os pixels brancos são associados ao ovo e os pixels pretos são associados ao fundo. Para diferenciar entre um *ROI* com ovo vermelho e um *ROI* vazio, é feita uma análise da área da máscara. Ao avaliar diferentes imagens, obteve-se que a área máxima em pixels encontrada em um *ROI* vazio foi 1142, e a área mínima para ovos vermelhos foi 9286, portanto, *ROIs* com áreas maiores que 8000 são considerados ovos vermelhos, caso contrário, são classificadas como vazios.

Neste ponto, cada *ROI* classificado como um ovo branco ou vermelho tem associada uma máscara binária que é submetida a uma simplificação para se concentrar apenas na região do ovo e evitar o processamento desnecessário. Para fazer isso, um recorte da máscara do ovo é feito a partir das coordenadas do quadro delimitador do contorno, conforme mostrado na Figura 73.

Figura 73 – Recorte da região de interesse.



Fonte – O autor.

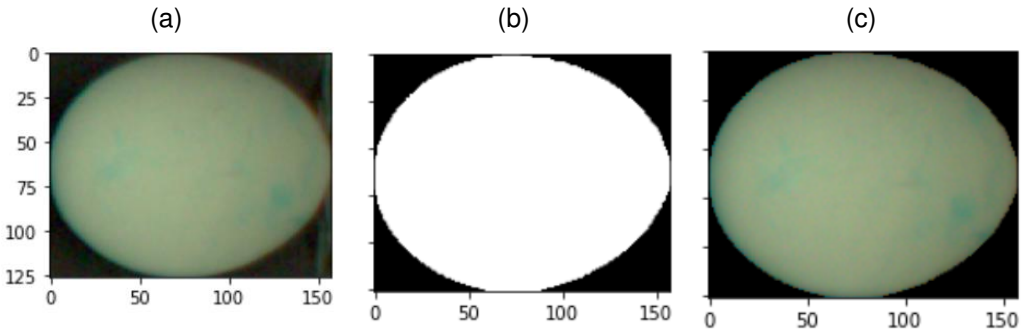
4.2.2 Algoritmo de detecção de sujeira

Dois algoritmos para detecção de sujeira foram desenvolvidos, um para processamento de ovos brancos e outro para ovos vermelhos, esses dois algoritmos são descritos a seguir:

a. *Algoritmo de detecção de sujeira em ovos brancos:* Este algoritmo tem como entrada a imagem previamente recortada no formato *BGR*, o cinza e a imagem binária. A imagem binária foi usada como uma máscara sobre as imagens aplicando a operação lógica *AND* entre duas imagens. Um exemplo desse processo na imagem *BGR* é mostrado na Figura 74.

   As manchas de sujeira nos ovos podem vir de diferentes origens, entre as mais comuns estão: manchas de fezes, ácido úrico, sangue e gema. Como essas manchas vêm em tons diferentes, é necessário criar máscaras diferentes que

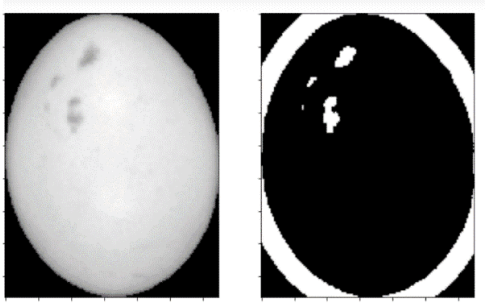
Figura 74 – (a) Imagem RGB; (b) Máscara; (c) Operação lógica *AND* entre (a) e (b).



Fonte – O autor.

permitam a localização desses defeitos. As manchas escuras são mais fáceis de detectar em ovos brancos porque apresentam um bom contraste. Para detectar este tipo de manchas, é realizada uma binarização com um limiar adaptativo na imagem cinza, esta binarização permite calcular um valor de limiar para regiões menores da imagem, neste caso o limiar é calculado a partir da média dos pixels da região avaliada. Ter diferentes limiares para cada região permite obter um melhor resultado quando se tem imagens com iluminação variável. Um exemplo do resultado dessa binarização adaptativa é apresentado na Figura 75.

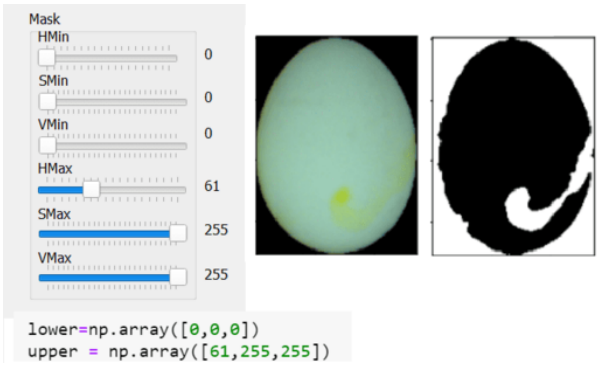
Figura 75 – Detecção manchas escuras em ovos brancos com binarização adaptativa.



Fonte – O autor.

As manchas com diferentes tonalidades requerem processamento adicional. Foi decidido trabalhar no espaço de cores *HSV*, pois permite uma melhor diferenciação das cores na imagem. Para realizar a segmentação das manchas é aplicada uma limiarização simples, onde é definido um limite inferior e um limite superior para cada um dos canais da imagem. Para encontrar esses limiares, foi utilizado um algoritmo disponível em (NATHANCY, 2020), no qual é possível avaliar diferentes combinações de valores de *H-S-V* de forma fácil e rápida a partir de *trackbars*. Usando este algoritmo, foi possível encontrar uma faixa de valores capaz de segmentar tons de manchas claras no ovo branco como mostrado na Figura 76.

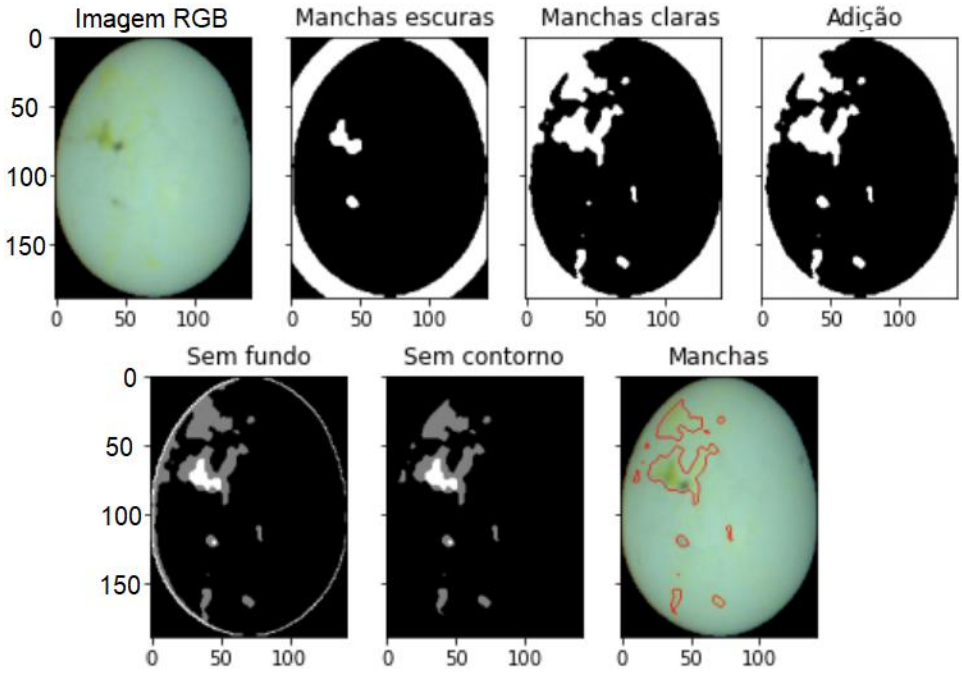
Figura 76 – Detecção manchas claras em ovos brancos.



Fonte – O autor.

Como um ovo pode apresentar um ou vários tipos de manchas, sejam claras ou escuras, foi realizada a integração dos dois processamentos anteriores, aplicando-se uma operação matemática de adição entre as duas imagens (claras e escuras), como o objetivo é detectar as manchas os pixels associados ao fundo e contorno foram retirados. Finalmente são obtidos apenas os pixels associados à sujeira, conforme mostrado na Figura 77.

Figura 77 – Detecção de manchas de sujeira em ovos brancos.

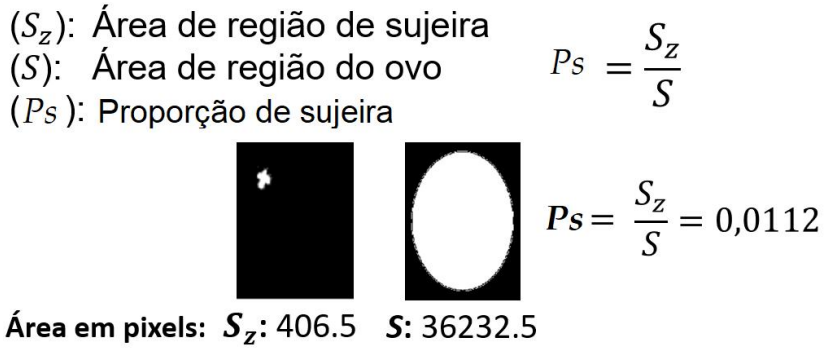


Fonte – O autor.

Para realizar a classificação, três componentes são analisados: a área total do ovo encontrada a partir da máscara do ovo, a área total da mancha encontrada dos pixels associados à sujeira final e uma proporção de sujeira *(Ps)* predefinida

como limite para a tomada de decisão. O parâmetro *Ps* é um valor obtido a partir da razão da área de sujeira máxima permitida para classificar um ovo como limpo sobre a área total do ovo. Um exemplo do cálculo dessa proporção é apresentado na Figura 78. Este parâmetro pode ser alterado a qualquer momento em relação ao tamanho da sujeira a ser descartado da linha.

Figura 78 – Exemplo da proporção *Ps* para classificação de sujeira.

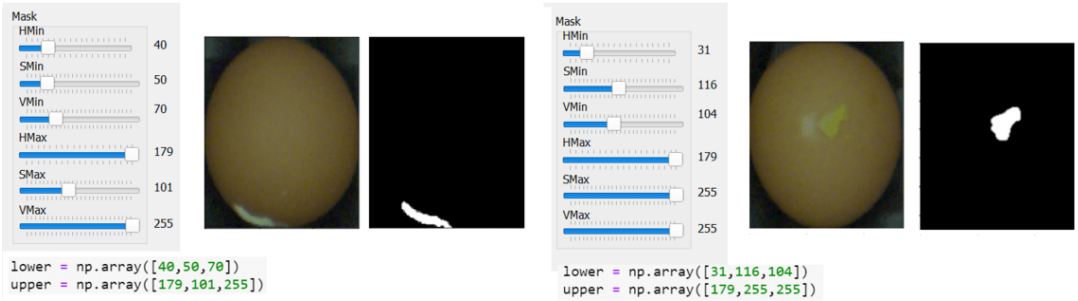


Fonte – O autor.

Um ovo é considerado sujo se a proporção *Ps* em questão (*PsA*) for maior ou igual à proporção *Ps* predefinida como parâmetro ($PsA \geq Ps$). Caso seja positivo, o ovo é classificado como sujo. Caso contrário, como limpo.

b. *Algoritmo de detecção de sujeira em ovos vermelhos:* Este algoritmo tem como entrada as imagens previamente recortadas nos formatos de imagem *BGR*, cinza e binário. Para detectar manchas claras associados a fezes e gemas, a imagem em formato *HSV* é binarizada usando dois faixas de valores de intensidade de pixel para cada canal. Os limites dessas faixas são mostrados na Figura 79.

Figura 79 – Detecção de manchas claras em ovos vermelhos.

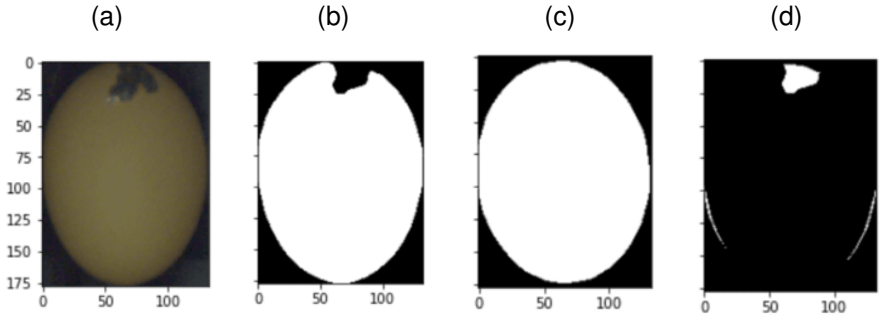


Fonte – O autor.

A detecção de manchas escuras representa um desafio quando se tem manchas com uma totalidade igual ao fundo que estão nas bordas do ovo, como é o caso da imagem mostrada na Figura 80a. Para enfrentar esse desafio, uma

aproximação elíptica do contorno da máscara binária é realizada para fechar as lacunas causadas pelas manchas (Figura 80c). Em seguida, é realizada uma operação de subtração matemática entre as duas imagens binárias, obtendo-se os pixels associados às manchas escuras. (Figura 80d).

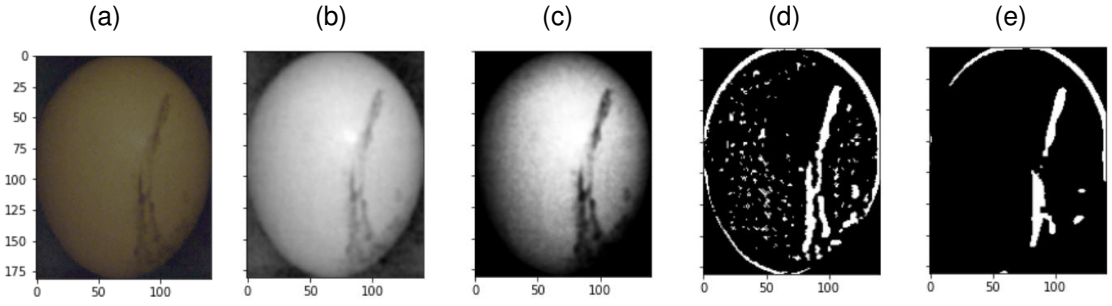
Figura 80 – (a) Imagem RGB; (b) Máscara; (c) Aproximação elíptica; (d) Subtração de (b) e (c).



Fonte – O autor.

Outra das manchas que representa um desafio são as manchas escuras com tonalidades próximas ao marrom, pois são muito semelhantes à tonalidade do ovo. Neste processo é utilizada a imagem no formato CINZA, o contraste da imagem é aumentado usando a equalização do histograma (Figura 81c). Após é realizada uma binarização adaptativa (Figura 81d), é por último, uma operação de filtragem para eliminar possíveis ruídos (Figura 81e).

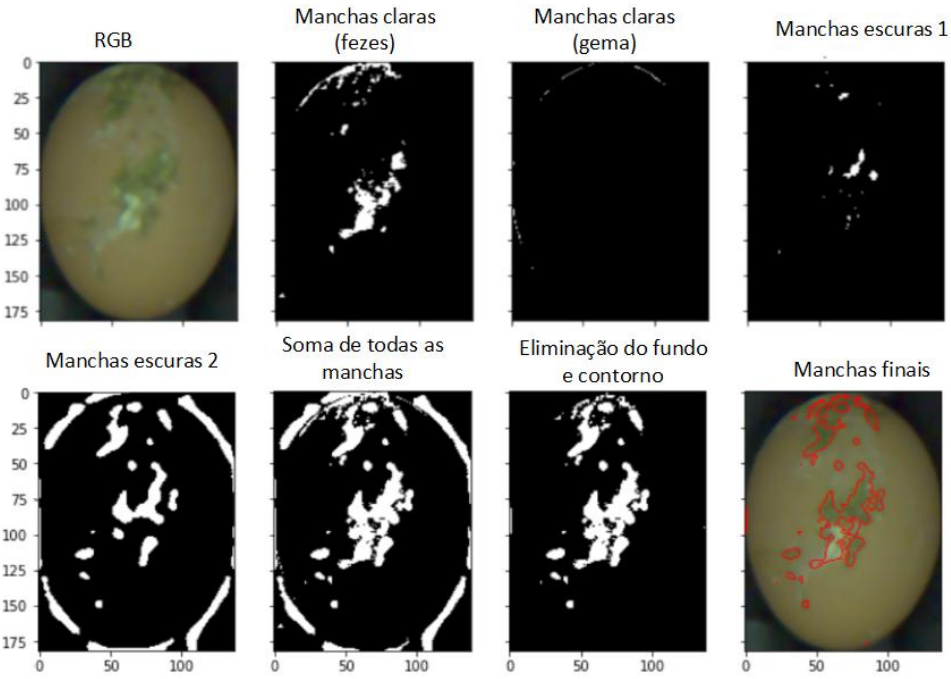
Figura 81 – (a) Imagem RGB; (b) Imagem cinza; (c) Equalização do histograma; (d) Binarização adaptativa; (e) Imagem após remoção do ruído.



Fonte – O autor.

Para concatenar todas as manchas, é realizada uma adição matemática entre todas as imagens, o fundo e o contorno são removidos, obtendo-se as áreas associadas as possíveis manchas (Figura 82).

Figura 82 – Detecção de sujeira em ovos vermelhos.



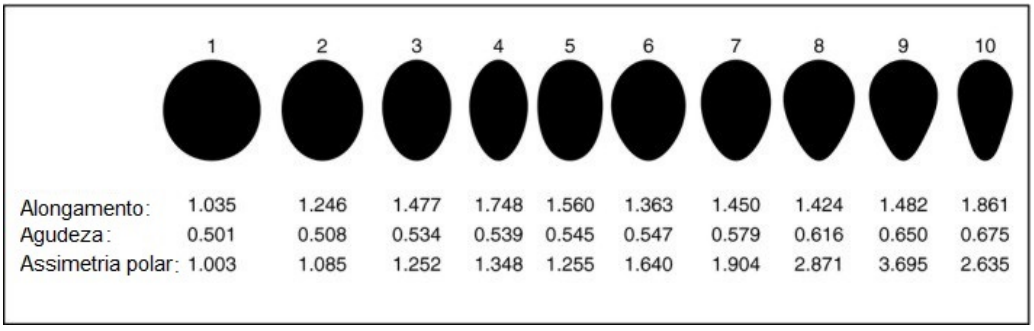
Fonte – O autor.

O processo de classificação da sujeira para ovos vermelhos é o mesmo que para ovos brancos onde são usados a área total do ovo obtida da máscara do ovo, a área total da sujeira final e o parâmetro *Ps*.

4.2.3 Algoritmo de formato

O algoritmo de formato foi aplicado nas máscaras binárias dos ovos. Para definir a forma do ovo, este algoritmo considera três índices: alongamento, agudeza e assimetria polar, definidos em (BIGGINS *et al.*, 2018). A Figura 83 mostra as 10 formas diferentes que um ovo pode ter de acordo com esses três índices.

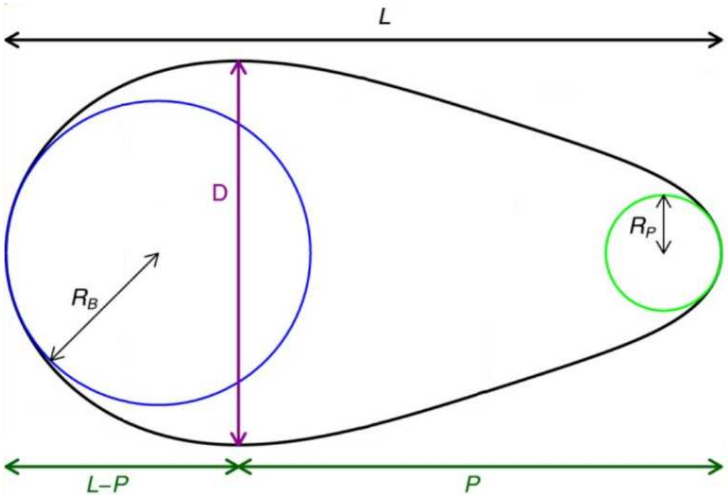
Figura 83 – Os valores dos três índices de forma para ovos de formas variadas.



Fonte – (BIGGINS *et al.*, 2018).

Esses índices foram escolhidos porque são fáceis de interpretar e refletem os aspectos biologicamente mais importantes da forma do ovo. Para determinar esses índices, é necessário considerar as medidas apresentadas na Figura 84. Onde *L* é o comprimento do ovo, *D* é o maior diâmetro latitudinal, *P* é a longitude da latitude de diâmetro maior até o polo mais distante; *RB* e *RP* são os raios dos maiores círculos dentro do ovo que tocam o polo agudo e pontiagudo, respectivamente.

Figura 84 – Medidas do formato do ovo.



Fonte – Editado de (BIGGINS *et al.*, 2018).

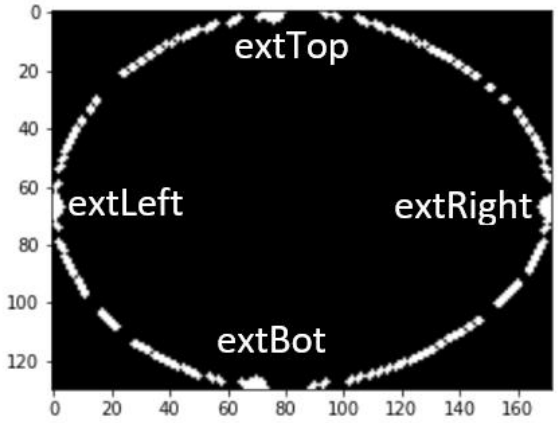
O alongamento é a relação entre o comprimento do maior diâmetro latitudinal (*D*) e o comprimento (*L*) do ovo. Porém, neste estudo, o alongamento foi definido como o recíproco do alongamento (*L/D*), de forma que os valores sejam sempre $\geq$1. Valores mais altos correspondem a maior alongamento. A agudeza é a longitude da latitude de diâmetro máximo até o polo mais distante (*P*) dividida pela longitude total (*L*). Esse índice está relacionado ao grau de assimetria do ovo, quanto mais pontudo mais assimétrico ele é. E a assimetria polar é a razão entre o diâmetro do maior círculo que pode caber dentro do contorno do ovo e tocar o ovo em seu polo rombudo (R_B) com o diâmetro do maior círculo dentro do contorno do ovo que toque no polo mais pontiagudo (R_p). Este índice permite definir a simetria do ovo, levando em consideração que ambas as pontas podem ser pontiagudas ou redondas. Os maiores valores desses índices correspondem a maiores desvios de um formato circular. O seguinte é uma descrição passo a passo do processo de encontrar o formato do ovo utilizando a visão computacional.

- Rotação da imagem, um aspecto fundamental na avaliação dos índices definidos em (BIGGINS *et al.*, 2018) é a posição do ovo, que deve ser horizontal para garantir a validade das equações. Portanto, o primeiro passo é avaliar as dimensões da imagem, caso o comprimento vertical fosse maior que o comprimento horizontal,

a imagem é girada 90° em torno do eixo *x* no sentido anti-horário, caso contrário, a imagem permanece inalterada.

- Determinar os pontos extremos ao longo do contorno, para encontrar os pontos extremos do ovo são utilizadas as operações definidas em (ROSEBROCK, 2016a). Esses pontos são mostrados na Figura 85.

Figura 85 – Pontos extremos ao longo do contorno do ovo.

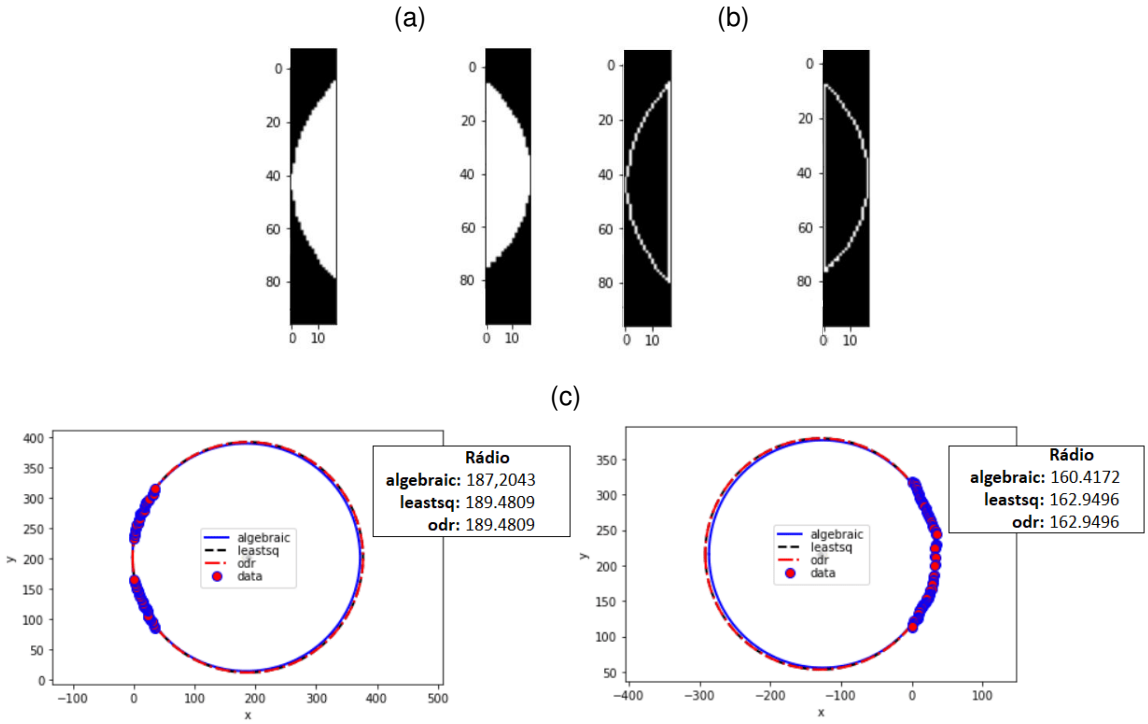


Fonte – O autor.

- Encontrar os parâmetros *D*, *L* e *P*, esses parâmetros são encontrados a partir dos pontos extremos do contorno do ovo, onde *D* é encontrado da subtração do valor em *y* dos pontos verticais, *L* a subtração do valor em *x* nos pontos horizontais e *P* é o maior comprimento entre a subtração do ponto em *x* do ponto superior ou inferior menos o valor *x* dos pontos à direita e à esquerda.

- Encontrar os raios R_B e R_P, esses raios são encontrados aplicando uma operação de mínimos quadrados no conjunto de pontos em *2D (x, y)* dos contornos associados aos polos do ovo. Este processo é dividido em 4 etapas, primeiro, uma área de 5 mm de largura é selecionada de ambos os lados da imagem do ovo (Figura 86a). Esta largura foi selecionada com base nos testes realizados em (VASILEVA *et al.*, 2018). Para definir a proporção do número de pixels por uma determinada métrica, o algoritmo é baseado no método proposto em (ROSEBROCK, 2016b), usando um objeto circular de 90 mm como referência, o resultado foi de 3.458 pixels por milímetro. Usando essa relação, é possível calcular o tamanho dos objetos em uma imagem. Depois, são obtidos os vetores associados aos contornos dos polos do ovo (Figura 86b). Para selecionar o método de mínimos quadrados mais eficiente, foram avaliados três métodos propostos em (ELBY, 2011) os quais foram: *algebraic approximation (algebraic)*, *Optimized least squares (leastsq)* e *Orthogonal Distance Regression* (*ODR*). Obtendo como resultado os raios mostrados na Figura 86c.

Figura 86 – (a) Área de 5 mm de largura de ambos os lados da imagem do ovo; (b) Contorno da imagem (a); (c) Operação de mínimos quadrados do contorno (b).



Fonte – O autor.

Por não ter uma variação considerável entre os resultados, as opiniões do autor foram consideradas na pesquisa, onde se constatou que tanto *ODR* quanto *leastsq* obtiveram os mesmos resultados, porém *leastsq* é o método mais eficiente e pode ser duas a dez vezes mais rápido que o *ODR*. Por outro lado, *algebraic* não é altamente recomendada para este tipo de aplicação, pois é limitada quando há apenas um arco para ajustar. Portanto, optou-se por aplicar *leastsq* usando a função *jacobian*, pois, de acordo com o autor, isso pode levar à redução do número de chamadas de função por um fator de dois a cinco. Uma vez encontrados os raios associados aos contornos dos polos esquerdo e direito (*RI* e *RD*), estes são comparados para definir o maior (R_B) e o menor (R_P).

- Encontrar os índices de formato, uma vez obtidos os parâmetros *D, L, P,* R_B e R_P, são encontrados os índices de alongamento, agudeza e assimetria polar.

Para realizar a classificação, foi definido um vetor de referência de cada formato apresentado na Tabela 4. Cada nova amostra é classificada por meio de um classificador por distância mínima, onde o vetor avaliado é comparado com todos os 10 vetores associados a cada formato, em cada comparação é calculada a distância euclidiana entre os dois vetores. No final, obtém-se um vetor com 10 valores onde a posição do

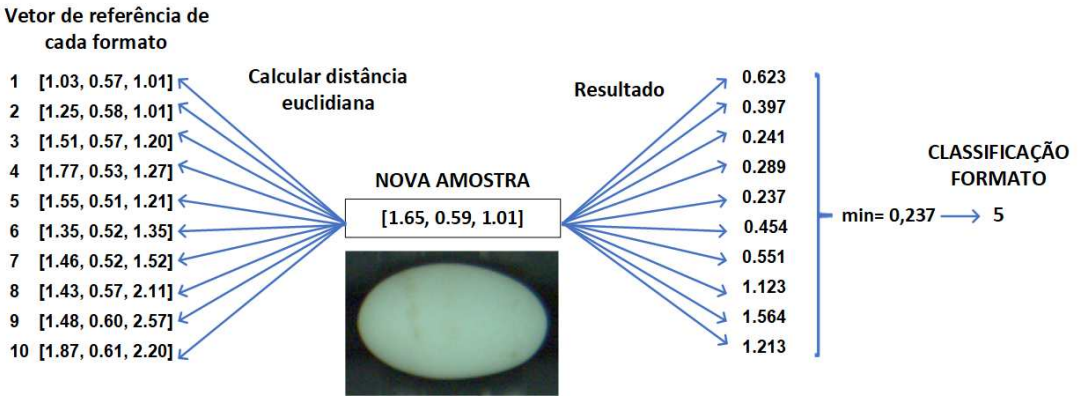
vetor em que o valor é mínimo corresponde ao formato do ovo. Um exemplo desse processo é mostrado na Figura 87.

Tabela 4 – Vetor de referência de cada formato do ovo.

Formato	Elongação	Agudeza	Assimetria Polar
1	1,028986	0,570423	1,010563
2	1,254386	0,58042	1,01072
3	1,505263	0,56993	1,201776
4	1,765432	0,534965	1,272414
5	1,554348	0,51049	1,213416
6	1,352381	0,524648	1,347607
7	1,459184	0,520979	1,524208
8	1,43	0,56993	2,112691
9	1,479167	0,602113	2,566391
10	1,868421	0,612676	2,204685

Fonte – O autor.

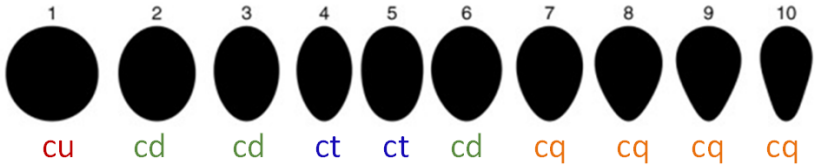
Figura 87 – Classificação do formato do ovo usando um classificador de distância mínima.



Fonte – O autor.

Para simplificar a classificação da geometria do ovo a partir dos índices de forma, foi realizada a união de alguns formatos que apresentavam semelhança, por exemplo, os formatos 2, 3 e 6 apresentados na Figura 83 são os que mais se aproximam de uma forma normal de ovo por apresentarem simetria entre os seus polos, os formatos 4 e 5 têm a característica de possuírem ambos os polos agudos ou ambos os polos pontiagudos, os formatos 7, 8, 9, 10 caracterizam-se por serem alongados com maior assimetria polar. Como resultado dessa união, quatro categorias de formato de ovo são obtidas para serem classificadas: categoria 1 (cu) - forma 1, categoria 2 (cd) - formatos 2, 3 e 6, categoria 3 (ct) - formatos 4 e 5, e categoria 4 (cq) - formatos 7, 8, 9 e 10, conforme mostrado na Figura 88.

Figura 88 – Agrupamento dos formatos do ovo.



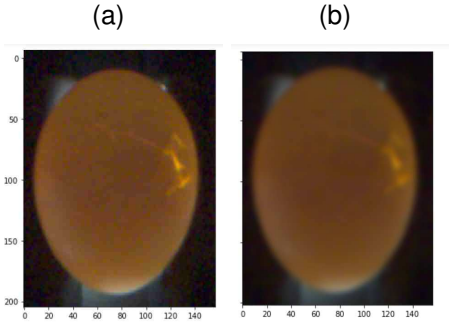
Fonte – O autor.

Finalmente, para realizar a classificação, considera-se o resultado obtido pela classificação da distância mínima e avalia-se se o formato obtido é aceitável ou inaceitável.

4.2.4 Algoritmo de detecção de fissuras

Este algoritmo tem como entrada os *ROIs* das imagens capturadas com iluminação inferior. O algoritmo de detecção de fissuras é semelhante entre as duas cores do ovo. Inicialmente, o brilho da imagem é reduzido e uma operação de filtragem é aplicada para suavizar a imagem (Figura 89), isso é feito com o objetivo de atenuar o ruído produzido pelas manchas escuras existentes na superfície do ovo que são mais visíveis quando a luz passa pela casca.

Figura 89 – (a) Imagem original; (b) Imagem suavizada.

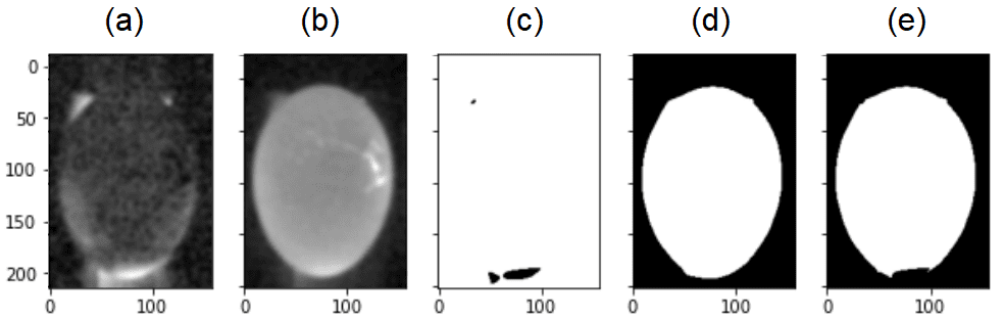


Fonte – O autor

Posteriormente, os componentes *B* e *R* são extraídos da imagem. O componente *B* é usado para encontrar possíveis reflexos de luz que podem ser confundidos com rachaduras na casca e o componente *R* para determinar a máscara do ovo. Uma binarização simples é aplicada às imagens com os limites [80-255] e [50-255] para ambos os componentes em ovos brancos e ovos vermelhos respectivamente. As imagens binárias obtidas são multiplicadas a fim de se obter a máscara de ovo sem reflexos de luz conforme mostrado na Figura 90.

Uma vez adquirida a máscara do ovo, é aplicada ao componente *R* da imagem para analisar apenas a região associada ao interior do ovo. Para encontrar as fissuras,

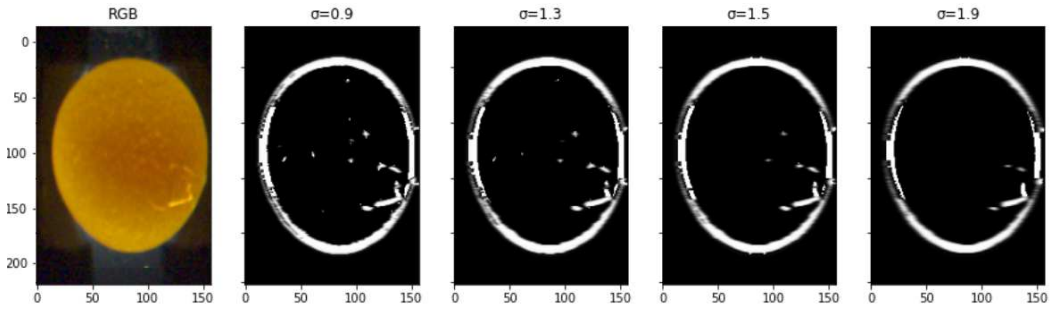
Figura 90 – (a) Componente *B*; (b) Componente *R*; (c) Imagem binária associada ao brilho; (d) Máscara do ovo; (c) Multiplicação de (c) e (d).



Fonte – O autor.

é realizado um processo de detecção de bordas usando um filtro Laplaciano Gaussiano. Para definir o valor de σ no filtro gaussiano, foram realizados alguns testes com diferentes σ (Figura 91), isto com o objetivo de encontrar um valor que identifique as fissuras sem perder qualidade e ao mesmo tempo afastando o ruído causado pelas manchas escuras associadas à casca do ovo. Como resultado, obteve-se que um σ=1.9 permite a identificação de fissuras minimizando o ruído.

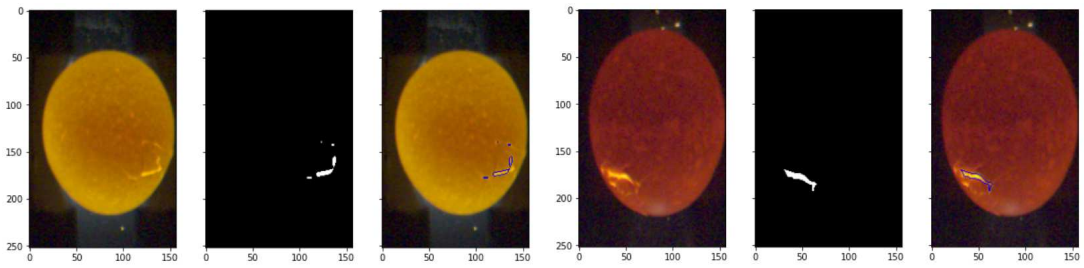
Figura 91 – Avaliação de diferentes valores de σ no filtro Laplaciano Gaussiano.



Fonte – O autor.

Finalmente a borda é removida e pequenos ruídos são eliminados com operações de filtragem, obtendo-se os contornos associados às fissuras (Figura 92).

Figura 92 – Detecção de fissuras em ovos brancos e vermelhos.



Fonte – O autor.

Para determinar se um ovo está fissurado, uma análise de contorno é realizada onde ovos com áreas maiores ou iguais a 15 foram considerados como fissurados.

4.3 ALGORITMOS DE CLASSIFICAÇÃO DE OVOS USANDO TÉCNICAS DE APRENDIZADO PROFUNDO

A fim de classificar os ovos nas quatro categorias estabelecidas na lista de requisitos funcionais do sistema, foi adicionalmente proposta a utilização de técnicas de aprendizado profundo. Para este efeito, são implementados dois métodos, classificação de imagens e segmentação semântica. Esses métodos foram implementados usando a biblioteca *Fastai* versão 2.1.8. O treinamento dos modelos foi realizado na plataforma *Google Colaboratory* carregando os dados do *Google Drive*. A descrição detalhada de sua implementação é apresentada a seguir:

4.3.1 Classificação de imagens

A primeira etapa da classificação de imagens consiste em definir o conjunto de dados para o treinamento e validação. Para isso foram selecionadas 29.493 imagens com ovos e 9.000 exemplos de imagens vazias em caso de não ter um ovo no rolo. As imagens tinham resolução de 157x252 pixels, as quais foram agrupadas nas seguintes classes:

- *Formato do ovo:* as quatro classes *cu, cd, ct* e *cq* usadas no algoritmo de formato com técnicas clássicas da seção 4.2.3, nas quais as 10 formas básicas que um ovo pode apresentar são agrupadas de acordo com os índices de alongamento, agudeza e assimetria polar;
- *Cor:* ovos brancos (b) e ovos vermelhos (v);
- *Defeitos de sujeira*: para cumprir o segundo requisito funcional do presente trabalho, decidiu-se criar três classes, ovos limpos (*l*), ovos levemente sujos (*ps*) e ovos muito sujos (*ms*), o que permite ter dois graus de sujeira nos ovos. Para classificar essas categorias, foram consideradas as informações do Quadro 6 em que ovos com um grau de sujeira que cobre menos de 1/32 da superfície do ovo são considerados limpos e ovos com um grau de sujeira que cobre mais de 1/4 da superfície do ovo como muito sujos;
- *Defeitos de fissuras*: fissurados (f) e sem fissura (sf).

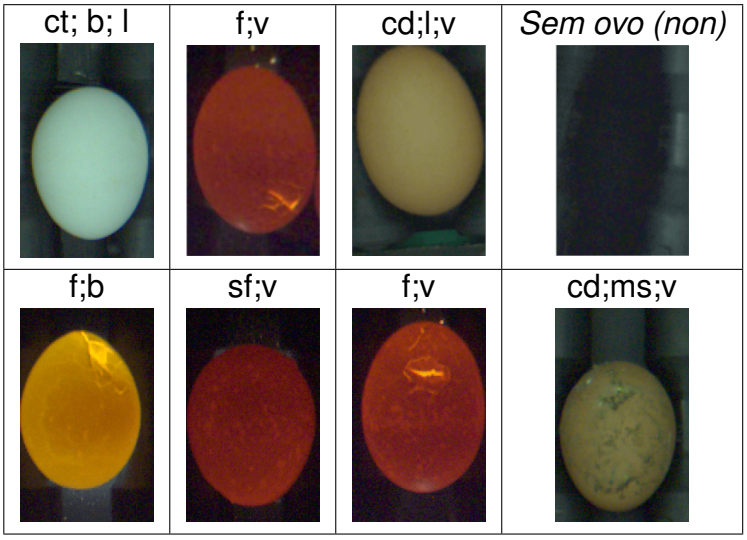
O total de imagens utilizadas para treinamento e validação dos modelos com suas respectivas classes é apresentado na Tabela 5. O conjunto de validação foi o 30 % do total de imagens em cada categoria. Alguns exemplos dessas imagens são mostrados no Quadro 7.

Tabela 5 – Classificação das Imagens de treinamento e validação dos modelos.

Fissuras	**Formato**	**Sujeira**	**Brancos**	**Vermelhos**
-	cu	ms	29	368
-	cu	l	307	1191
-	cu	ps	63	134
-	cd	ms	207	1435
-	cd	l	9728	9416
-	cd	ps	2403	1696
-	ct	l	134	10
-	ct	ps	60	9
-	cq	l	18	-
-	cq	ps	453	3
f	-	-	218	220
sf	-	-	255	270

Fonte – O autor.

Quadro 7 – Amostra de imagens de treinamento e validação dos modelos.



A segunda etapa consiste na realização do treinamento dos modelos. Para este trabalho, quatro modelos foram treinados com as arquiteturas *ResNet18*, *ResNet34*, *ResNet50* e *GoogLeNet*. Esses modelos foram pré-treinados no conjunto de dados de *ImageNet* e treinados no conjunto de dados criado para este projeto usando um tamanho do batch de 64. O conjunto de dados foi sujeito a normalização usando as estatísticas de dados de *ImageNet*. Além disso, foi feito um aumento de dados aplicando uma inversão espelho nas imagens. A perda de entropia cruzada foi usada como função da perda, *weight decay* como técnica de regularização e a acurácia, *precisão, Recall e F1Score* como métricas de avaliação.

As imagens e seus rótulos foram associados por meio de um arquivo *.csv* contendo três colunas: *fname*, *labels*, *is_valid* conforme mostrado na Figura 93. Onde *fname* corresponde ao nome de cada uma das imagens, *labels* corresponde ao rótulo

de cada imagem, neste caso, é realizada uma classificação *multi-tag*, e a última coluna *is_valid* define se a imagem corresponde ao grupo de treinamento (*False*) ou ao grupo de validação (*True*).

Figura 93 – Arquivo .csv para criação do *DataLoaders*.

	fname	labels	is_valid
0	13956.png	non	False
1	13957.png	non	False
2	13958.png	non	False
3	13959.png	non	False
...	...	...	...
38490	37498.png	v sf	True
38491	37499.png	v sf	True
38492	37500.png	v sf	True

38493 rows × 3 columns

Fonte – O autor.

Como primeira etapa da aprendizagem por transferência, é realizado o treinamento das duas últimas camadas criadas com valores aleatórios para o treinamento do conjunto de dados de referência, com todas as demais camadas congeladas. O tamanho dessas duas camadas depende do tamanho da saída da última camada das arquiteturas implementadas e do tamanho do número de classes a serem classificadas pelos modelos. Foram avaliadas diferentes épocas e taxas de aprendizagem obtendo os resultados apresentados na Tabela 6.

Tabela 6 – Resultados da primeira etapa do treinamento dos modelos de classificação de imagens.

Arquitetura	**Número de épocas**	**Taxa de aprendizagem**	**Acurácia**	**Precisão**	***Recall***	***F1Score***
ResNet18	2	2e-3	0,98479	0,86316	0,78851	0,81074
ResNet34	2	2e-3	0,98492	0,86464	0,78310	0,80349
ResNet50	2	2e-3	0,98727	0,89516	0,84079	0,85835
GoogleNet	2	2e-3	0,98265	0,85854	0,74902	0,78151

Fonte – O autor.

A segunda etapa da aprendizagem por transferência consiste em descongelar toda a rede e fazer o treinamento novamente. Ao avaliar diferentes épocas de treinamento e taxas de aprendizagem, os melhores resultados foram obtidos com os parâmetros apresentados na Tabela 7. Para selecionar o número de épocas de treinamento, foi considerada a curva de aprendizado dos modelos, onde como critério de

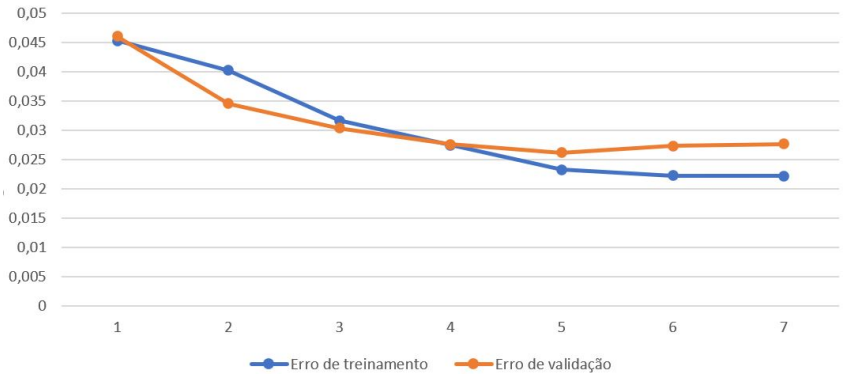
parada foi considerado o momento em que o erro no grupo de validação estagnou ou começou a aumentar. Este processo foi realizado para todos os modelos, obtendo-se bons resultados com apenas 5 épocas por ter modelos pré-treinados. Um exemplo da curva de aprendizado do modelo ResNet34 é apresentado na Figura 94 onde por mais de cinco épocas de treinamento, o erro no conjunto de validação começa a aumentar.

Tabela 7 – Resultados do treinamento dos modelos de classificação de imagens.

Arquitetura	**N.º de épocas**	**Taxa de aprendizagem**	**Técnica de regularização (*weight decay*)**	**Acurácia**	**Precisão**	***Recall***	***F1Score***
ResNet18	5	1,5e-6, 1,5e-4	λ= 2,9e-3	0,9883	0,9165	0,8428	0,8732
ResNet34	5	1e-6, 1,5e-3	λ= 2,59e-3	0,9900	0,9304	0,8877	0,9074
ResNet50	5	1,5e-5, 1,5e-3	λ= 2,5e-3	0,9908	0,9431	0,8956	0,9170
GoogleNet	5	1,5e-5, 1,5e-3	λ= 2,8e-3	0,9881	0,9281	0,8441	0,8787

Fonte – O autor.

Figura 94 – Curva de aprendizado de ResNet34



Fonte – O autor.

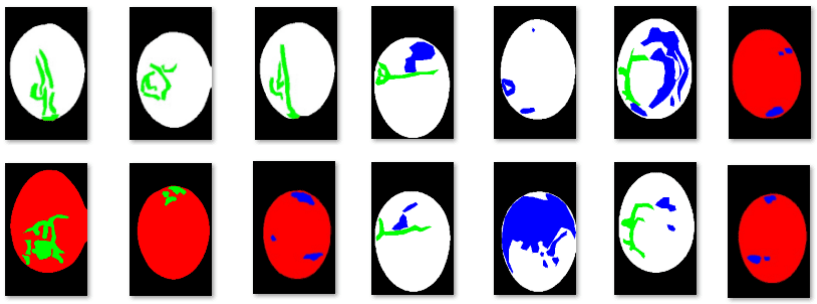
O modelo que obteve as maiores métricas foi o *ResNet50* com uma acurácia de 99 % e um F1-score de 91,7 %. Os quatro modelos foram selecionados para serem avaliados em um grupo de teste, a fim de determinar seu desempenho e tempo de processamento. Os resultados obtidos são apresentados no Capítulo 5.

4.3.2 Segmentação semântica

O segundo método de aprendizado profundo implementado para classificar as categorias dos ovos é a segmentação semântica. A primeira etapa é definir o conjunto de dados para treinamento e validação, no qual foram selecionadas 7.702 imagens com resolução 157x252 pixels, essas imagens foram submetidas a um processo de etiquetagem utilizando uma ferramenta de anotação de imagens gráficas conhecida como *Labelme* (TORRALBA *et al.*, 2010). Um exemplo dessas imagens é apresentado na Figura 95, onde o fundo é representado em preto, o ovo vermelho em vermelho,

o ovo branco em branco, as manchas de sujeira em azul e as fissuras em verde. O conjunto de dados foi dividido em 70 % para treinamento e 30 % para validação.

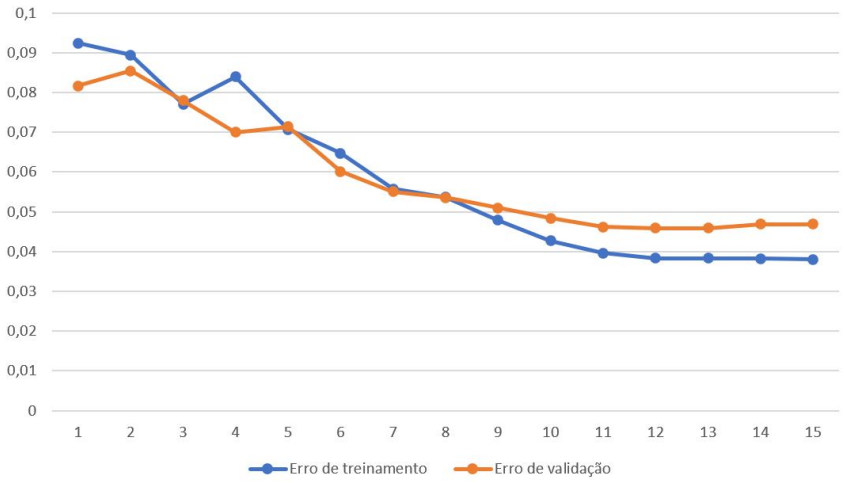
Figura 95 – Rotulagem de imagens para treinamento dos modelos de segmentação semântica.



Fonte – O autor.

Na segunda etapa, é realizado o treinamento dos modelos. Neste método foi utilizada a arquitetura *Unet* tendo como *backbone* as arquiteturas *ResNet18*, *ResNet34*, *ResNet50* e *GoogLeNet*. Os modelos foram pré-treinados no conjunto de dados de *ImageNet*. O tamanho do lote foi definido experimentalmente, foram avaliados diferentes tamanhos nos quais se constatou que o tamanho máximo possível que poderia ser utilizado para o treinamento era 12 devido ao fato de que o treinamento destes modelos requer grande capacidade de memória e a plataforma gratuita do *Google Colaboratory* fornece apenas 12,72 GB de memória. Para definir o número de épocas de treinamento, também foram analisadas as curvas de aprendizado de cada modelo. A Figura 96 mostra o exemplo da curva obtida no treinamento do modelo *Unet-ResNet18*, onde fica evidenciado que por mais de 13 épocas o erro no conjunto de validação começa a aumentar.

Figura 96 – Curva de aprendizado de *Unet-ResNet18*.



Fonte – O autor.

O conjunto de dados foi normalizado com as estatísticas de dados *ImageNet* e a inversão espelho foi realizada nas imagens para aumentar o conjunto de treinamento e validação. A perda de entropia cruzada foi usada como função de perda, o *weight decay* como função de regularização e *IoU* como métrica de avaliação. A métrica *IoU* utilizada foi obtida de (ECKELS, 2020), onde a média de *IoU* é calculada para todas as classes. Ao realizar aprendizagem por transferência avaliando diferentes épocas de treinamento e taxas de aprendizagem, foram obtidos os resultados apresentados na Tabela 8.

Tabela 8 – Resultados do treinamento dos modelos de segmentação semântica.

Arquitetura	**Tamanho do lote**	**Número de épocas**	**Taxa de Aprendizagem**	**Técnica de regularização *(weight decay)***	**Perda de validação**	***IoU***
Unet-ResNet18	12	13	1e-5, 1e-3	λ= 2,5e-3	0,045909	0,6731
Unet-ResNet34	12	11	1e-5, 1e-3	λ= 2,5e-3	0,049945	0,6678
Unet-ResNet50	12	10	1e-5, 1e-3	λ= 2,5e-3	0,051727	0,6625
Unet-GoogLeNet	12	14	1e-5, 1e-3	λ= 3,1e-3	0,052023	0,6712

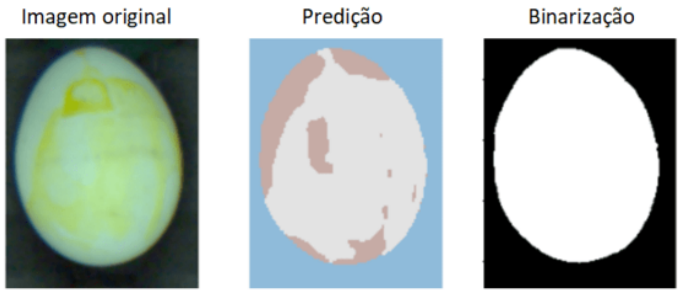
Fonte – O autor.

O modelo que apresentou maior *IoU* foi o *ResNet18* com um *IoU* de 67,31 % e perda no conjunto de validação de 0,045. Os quatro modelos foram selecionados para serem avaliados em um conjunto de teste para determinar seu desempenho e tempo de processamento. Os resultados obtidos são apresentados no Capítulo 5.

A última etapa deste método é adaptar os modelos de segmentação como algoritmos de classificação das quatro categorias do ovo definidas no Quadro 6. Essas categorias são as mesmas utilizadas no método de classificação de imagens definidas na seção anterior. Para realizar a classificação, são analisadas as previsões obtidas pelos modelos de segmentação. Essas previsões são tensores de tamanho 157x252 pixels, dos quais são obtidos os elementos únicos ordenados da matriz e as ocorrências de cada classe. A partir desses dois vetores é possível determinar a cor do ovo, os defeitos de sujeira, fissuras e imagens sem ovo. Para determinar o grau de sujeira, é encontrada a proporção entre a ocorrência de pixels com a classe associada à sujeira sobre a soma das ocorrências de pixels de todas as classes, exceto o fundo. Se a proporção for superior a 0,1 o ovo é classificado como muito sujo, se a proporção estiver entre 0,01 e 0,1 é classificado como pouco sujo, caso contrário, é classificado como limpo.

Neste algoritmo não é possível definir a forma do ovo a partir da segmentação, por isso é necessário aplicar o algoritmo clássico de formato proposto para este projeto. Este algoritmo recebe como entrada a máscara prevista pelo modelo previamente binarizada com os limites [1, 4]. A Figura 97 mostra um exemplo do resultado dessa binarização.

Figura 97 – Máscara do ovo usando segmentação semântica.



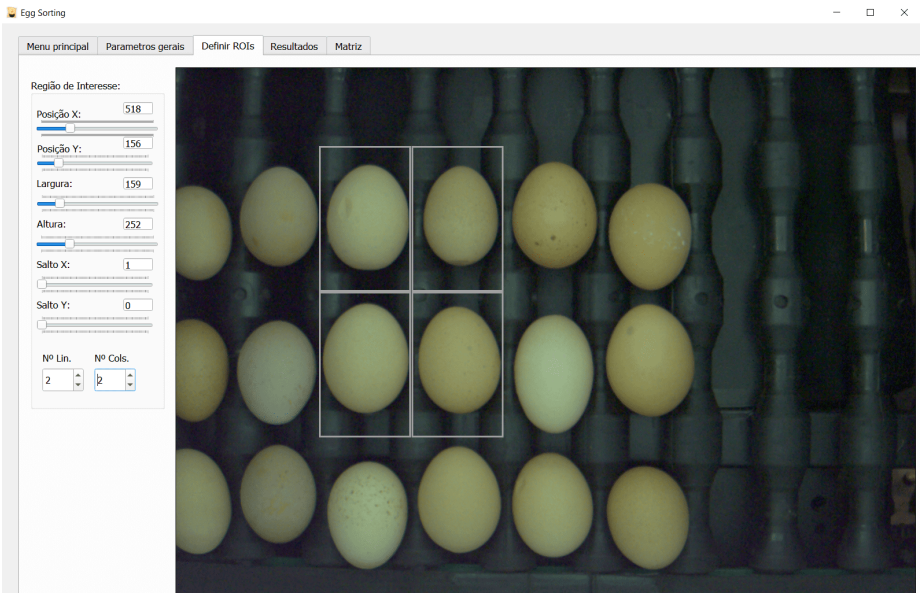
Fonte – O autor.

4.4 SOFTWARE DE PARAMETRIZAÇÃO DA INSPEÇÃO

Um sistema de software foi criado para separação de regiões de interesse, parametrização dos algoritmos empregados e facilitar a visualização dos resultados. Tal sistema, que possui uma interface gráfica para facilitar a utilização, foi implementado em C++ utilizando a biblioteca gráfica QT. As regiões de interesse e os parâmetros são definidos da seguinte forma:

a. *Regiões de Interesse (ROIs)*: são definidas as localizações dos ovos a serem analisados. Para simplificar a visualização dos resultados e a execução dos algoritmos, foi considerada apenas a captura de uma única câmera. Portanto, será possível definir de 1 a 18 *ROIs* para análise. Um exemplo de seleção de *ROIs* na interface desenvolvida é mostrado na Figura 98.

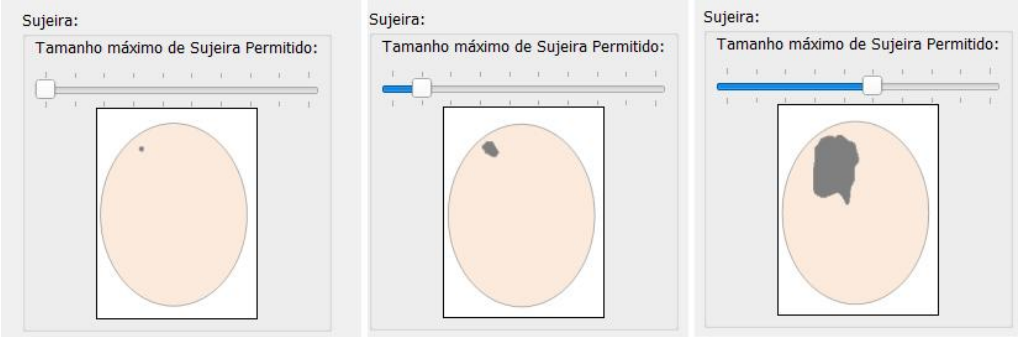
Figura 98 – Definição das regiões de interesse na interface.



Fonte – O autor.

b. *O grau de sujeira:* é possível definir o tamanho da mancha aceitável pelo sistema. A Figura 99 mostra como esse parâmetro é selecionado na interface. Esse tamanho será interpretado internamente como uma proporção (Ps) obtida a partir da relação entre a área da mancha e a área total do ovo.

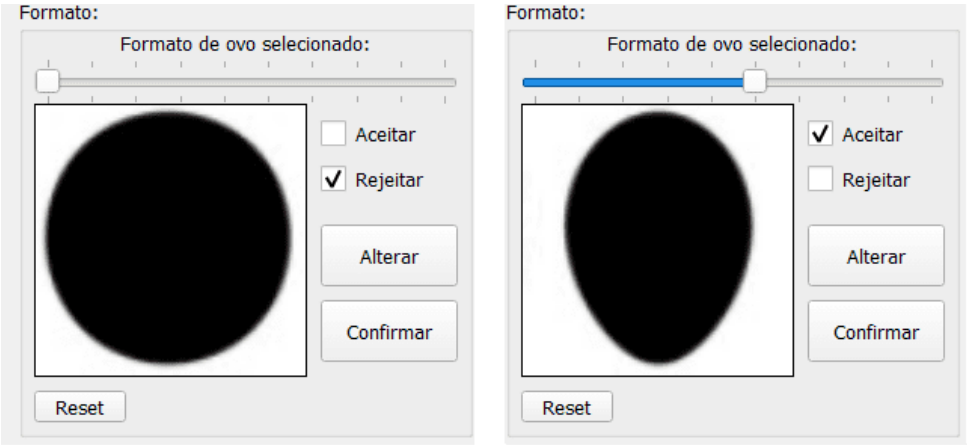
Figura 99 – Seleção do tamanho máximo de sujeira permitido na interface desenvolvida.



Fonte – O autor.

c. *O formato do ovo:* a partir deste parâmetro são definidas as geometrias dos ovos aceitáveis como normais e aquelas que devem ser rejeitadas, esses valores são armazenados em um vetor (VF) de 10 posições, onde True é igual ao formato aceitável e False formato inaceitável. Um exemplo de como a seleção do formato é feita na interface do usuário é mostrado na Figura 100.

Figura 100 – Seleção do formato do ovo na interface desenvolvida.



Fonte – O autor.

Mais detalhes sobre o design de interface do usuário proposto para esta pesquisa podem ser encontrados no apêndice D.

4.5 SUMÁRIO DO DESENVOLVIMENTO DO PROJETO

Neste trabalho, nove algoritmos apresentados na Tabela 9 foram propostos para classificar o ovo branco e vermelho nas categorias: normal, sujo, com geometria anormal e quebrado.

Tabela 9 – Sumário dos algoritmos propostos no projeto.

		Processamento	**Algoritmos de classificação**
Técnicas clássicas		Segmentação baseada em limiares, operações matemáticas, operações de filtragem, métodos de detecção de bordas e contagem de pixels para a tomada de decisões.	-Clássico
Técnicas de aprendizagem profunda	Classificação de imagem	Treinamento de arquitetura profunda CNN para realizar a classificação.	- *ResNet18* - *ResNet34* - *ResNet50* - *GoogLeNet*
	Segmentação semântica	Criação de uma base de imagem com etiquetagem manual dos pixels dos defeitos do ovo. Treinamento de uma arquitetura profunda para segmentação.	- *Unet-ResNet18* - *Unet-ResNet34* - *Unet-ResNet50* - *Unet-GoogLeNet*

Na próxima seção, são apresentados os resultados da avaliação desses algoritmos em um conjunto de teste, a fim de determinar seu desempenho na classificação. Os algoritmos são comparados para definir o algoritmo com melhor equilíbrio entre as métricas de avaliação e o tempo de processamento.

5 RESULTADOS

Este capítulo apresenta os resultados obtidos por cada algoritmo na classificação das quatro categorias de ovos definidas nos requisitos funcionais do sistema de inspeção. Em uma primeira etapa, são descritos os resultados individuais dos algoritmos clássicos e os algoritmos baseados em aprendizagem profundo. Em uma segunda etapa, é realizada uma análise conjunta de todos os algoritmos avaliados em um único conjunto de dados onde são analisadas as métricas de acurácia, precisão, *Recall*, *F1-score* e tempo de processamento.

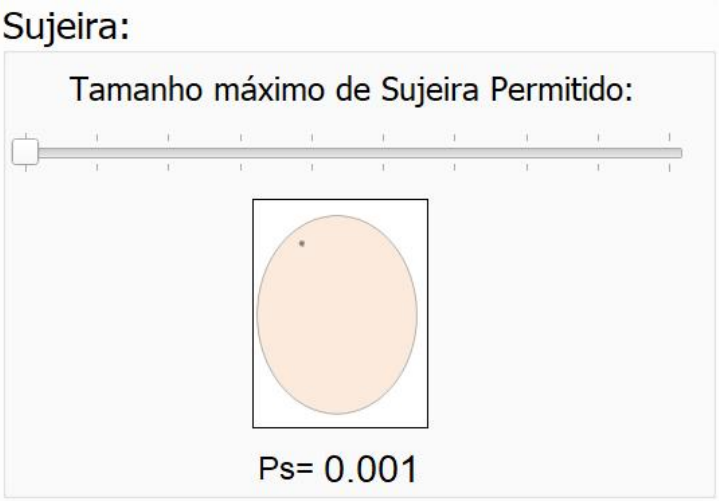
5.1 RESULTADOS DA CLASSIFICAÇÃO COM ALGORITMOS CLÁSSICOS

Esta seção descreve os resultados da avaliação dos algoritmos clássicos na detecção de sujeira, classificação de formato e detecção de fissuras.

5.1.1 Sujeira

Foram analisadas 4.313 imagens de ovos brancos e vermelhos, dos quais 1.610 tinham um grau de sujeira e 2.703 foram considerados limpos. Foi definida uma relação *Ps* de 0,001 para classificar os ovos (Figura 101). Nesse caso, a proporção *Ps* foi definida pequena para definir a precisão do algoritmo na detecção de qualquer tamanho de sujeira.

Figura 101 – Proporção *Ps* para classificação de sujeira.



Fonte – O autor.

A Tabela 10 mostra a matriz de confusão obtida pelo algoritmo de sujeira em ovos vermelhos e brancos. Da matriz de confusão é possível extrair as métricas de precisão, *Recall* e *F1-score* mostradas na Tabela 11. Nessa classificação, é analisado

o número de falsos negativos, ovos sujos que foram categorizados como limpos. Esses falsos negativos representam um grande problema, pois passariam para a linha de embalagem ovos que reduzem a qualidade do produto. Nesse caso, *Recall* é a métrica analisada quando há um alto custo associado a falsos negativos. Na Tabela 11 observa-se que em média o algoritmo obteve um *Recall* de 94,54 % na classificação de ovos sujos e de 93,60 % na classificação de ovos limpos. Nos resultados foi possível observar que a maioria dos falsos negativos se deu pelo fato de que na etapa de remoção do contorno foram eliminadas pequenas manchas que estavam muito próximas a ele. No entanto, este problema pode ser compensado quando o ovo gira 360° que permite observar a mancha em diferentes partes do ovo. Uma pequena amostra desses resultados é apresentada nos Quadros 8, 9 e 10.

Tabela 10 – Matriz de confusão obtida na classificação de sujeira.

		Vermelhos				**Brancos**	
		Predição				**Predição**	
		Sujos	**Limpos**			**Sujos**	**Limpos**
Real	**Sujos**	1053	41	**Real**	**Sujos**	479	37
	Limpos	95	1206		**Limpos**	77	1325

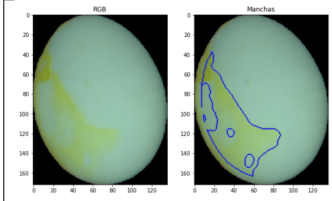
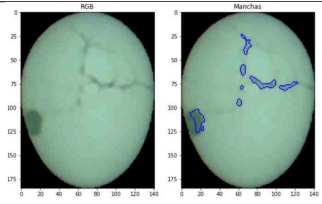
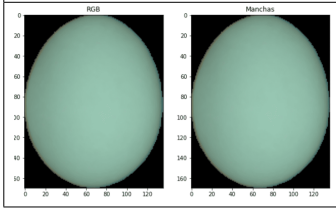
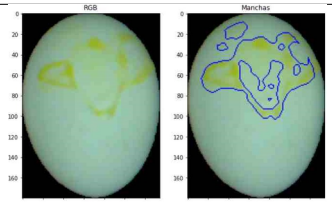
Fonte – O autor.

Tabela 11 – Métricas de avaliação na classificação de sujeira com algoritmos clássicos.

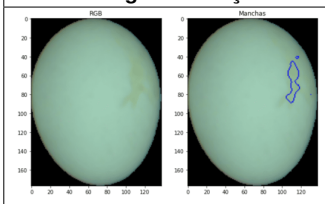
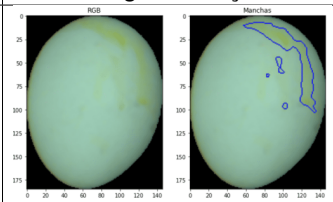
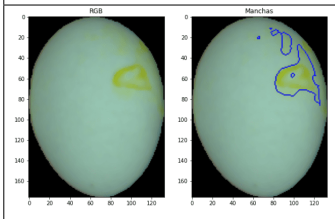
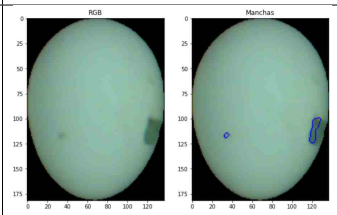
Métricas	**Sujos**		**Limpos**	
	Vermelhos	**Brancos**	**Vermelhos**	**Brancos**
Precisão	0,917	0,861	0,967	0,973
Recall	0,963	0,928	0,927	0,945
F1-score	0,939	0,894	0,947	0,959

Fonte – O autor.

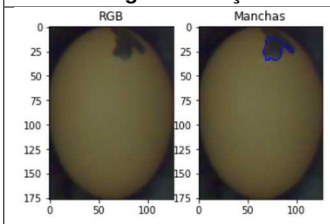
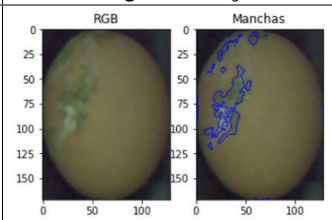
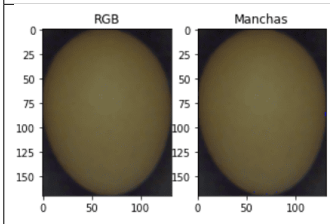
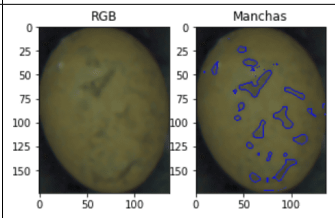
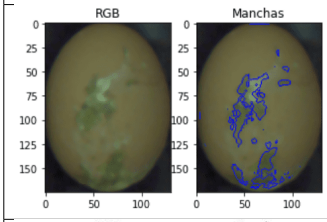
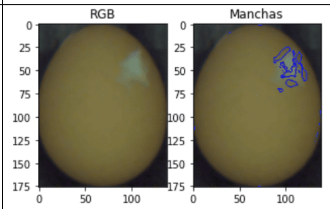
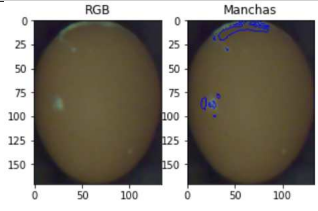
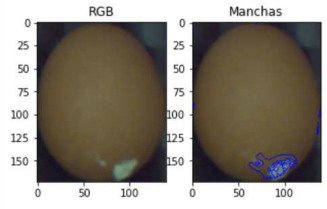
Quadro 8 – Resultados da detecção de sujeira em ovos brancos com algoritmos clássicos 1.

Imagem/Detecção	***PsA***	**Classificação**	**Imagem/Detecção**	***PsA***	**Classificação**
	0,2136	Sujo		0,0265	Sujo
	0,0	Limpo		0,3410	Sujo

Quadro 9 – Resultados da detecção de sujeira em ovos brancos com algoritmos clássicos 2.

Imagem/Detecção	*PsA*	Classificação	Imagem/Detecção	*PsA*	Classificação
	0,0164	Sujo		0,0756	Sujo
	0,0768	Sujo		0,007	Sujo

Quadro 10 – Resultados da detecção de sujeira em ovos vermelhos com algoritmos clássicos.

Imagem/Detecção	*PsA*	Classificação	Imagem/Detecção	*PsA*	Classificação
	0,0233	Sujo		0,0743	Sujo
	0,0	Limpo		0,0703	Sujo
	0,1048	Sujo		0,021	Sujo
	0,0241	Sujo		0,050	Sujo

De maneira geral, foi possível detectar manchas causadas por fezes, gema, sangue e penas com os algoritmos de sujeira para ovos vermelhos e brancos. O grau de sujeira definido pela proporção *Ps* permite ter uma variabilidade na sensibilidade para detecção de manchas, a grande vantagem deste parâmetro é que pode ser modificado a qualquer momento.

5.1.2 Formato

Um total de 11.942 imagens foram selecionadas e classificadas de acordo com as quatro categorias *cu, cd, ct* e *cq* definidas na seção 4.2.3 mostradas no Quadro 11. Cada imagem foi classificada em uma categoria usando a visão humana. A matriz de confusão e a análise das métricas obtidas na classificação do formato do ovo são apresentadas na Tabela 12. Neste caso, espera-se ter o menor número de falsos positivos e o menor número de falsos negativos. Para isso, é analisada a métrica *F1-score*, que permite determinar um equilíbrio entre precisão e *Recall*. Para o cálculo da média da métrica de *F1-score*, foi considerado o número de amostras de cada classe, obtendo-se um *F1-score* ponderado conhecido como *Micro-F1score*, nesta métrica, o algoritmo de classificação de formato obteve 96,7 %.

A categoria com maior número de falsos negativos entre as quatro foi *cd*, em que a maior parte desses números se deveu à classificação incorreta como *cu*. Esse caso pode ocorrer quando as categorias apresentam semelhanças entre elas, o que não permite a obtenção de índices significativamente diferenciáveis entre as formas.

Quadro 11 – Classificação de imagens de ovos por formatos.

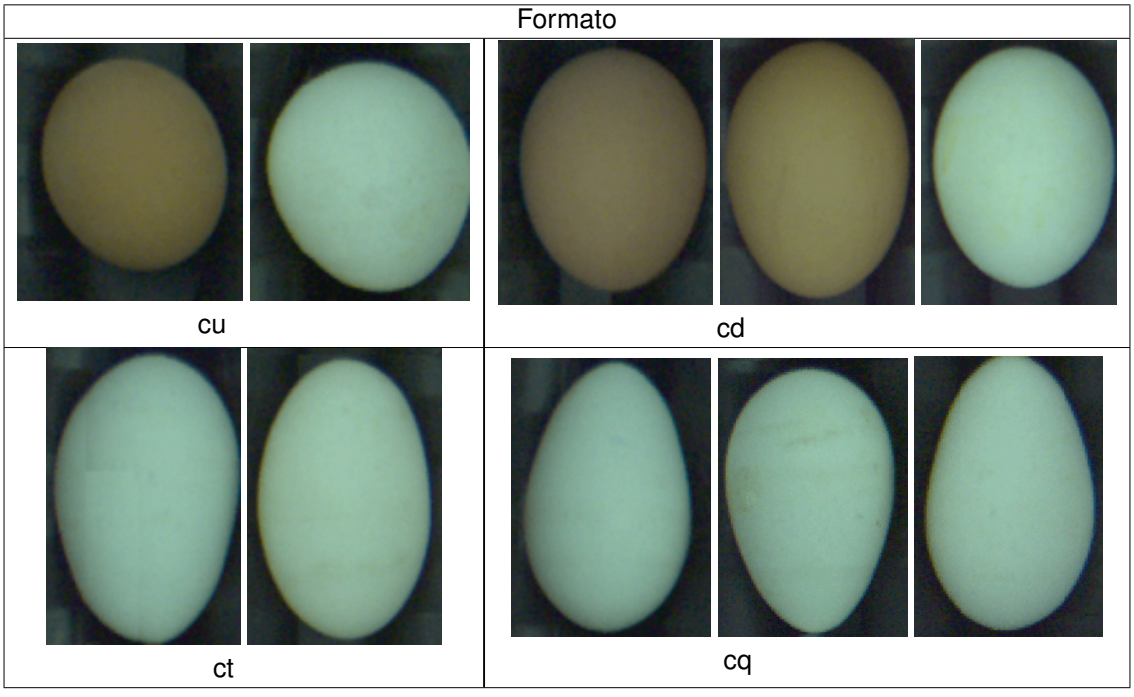


Tabela 12 – Resultados da classificação de formato usando algoritmos clássicos.

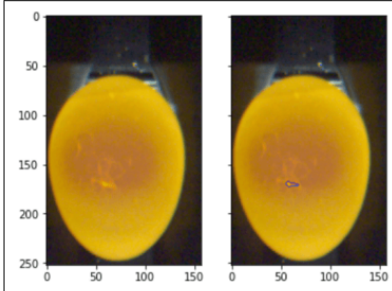
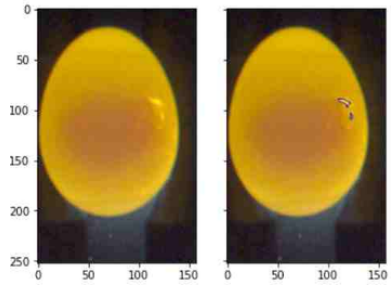
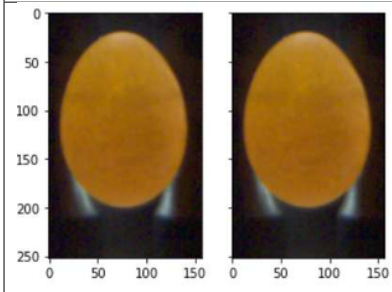
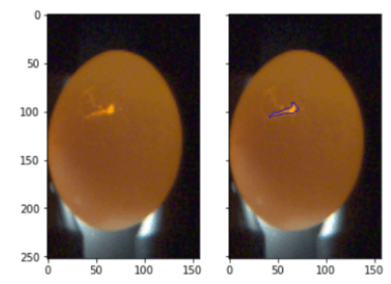
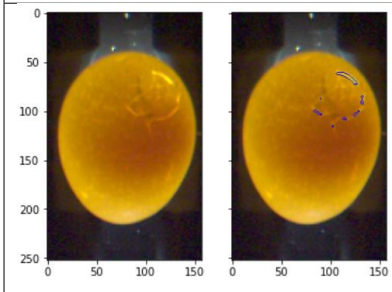
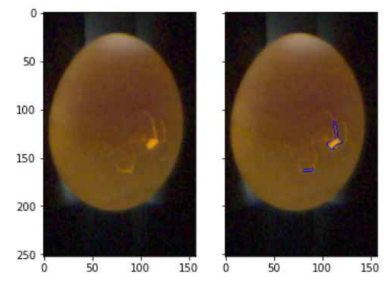
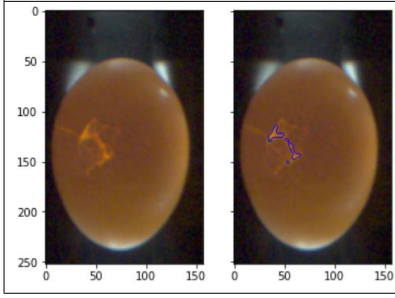
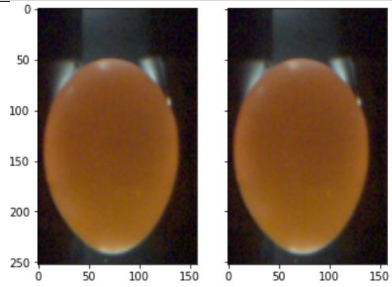
		Predição do algoritmo de formato				Precisão	*Recall*	*F1-score*
		cu	**cd**	**ct**	**cq**			
Real	**cu**	1778	123	0	4	0,898	0,933	0,915
	cd	202	9203	2	13	0,983	0,977	0,980
	ct	0	5	166	11	0,943	0,912	0,927
	cq	0	33	8	394	0,934	0,906	0,919

A partir dos resultados, pode-se afirmar que o uso do algoritmo de formato permite uma classificação das formas dos ovos usando índices de forma com uma precisão média de 93,9 % e um *Recall* de 93,2 %. A eficiência deste algoritmo depende fortemente da correta segmentação da máscara binária associada ao ovo na imagem.

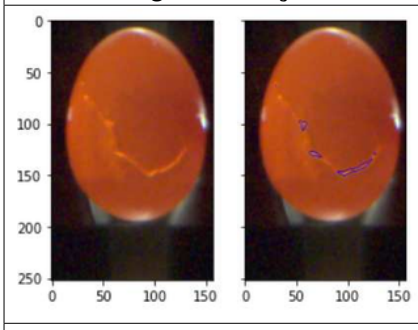
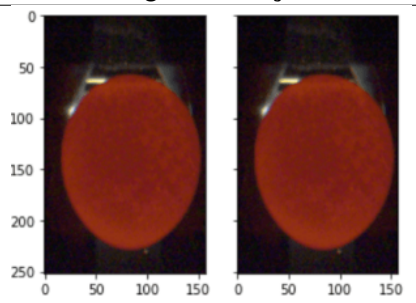
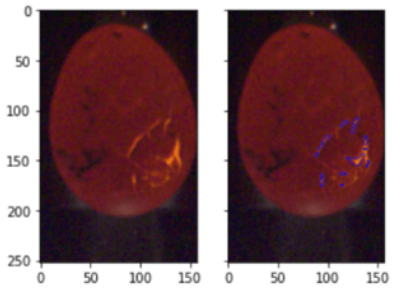
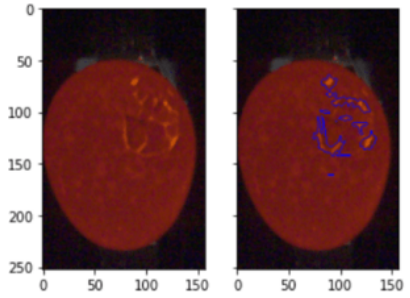
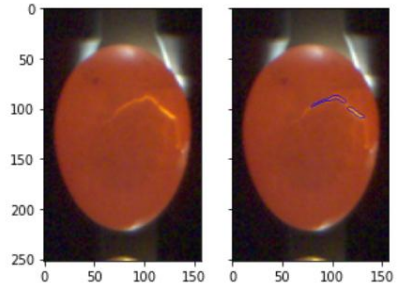
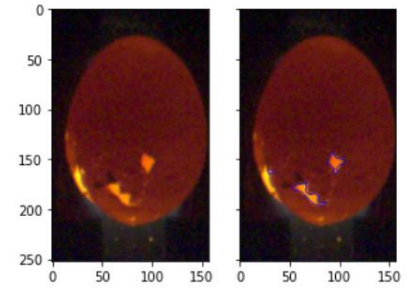
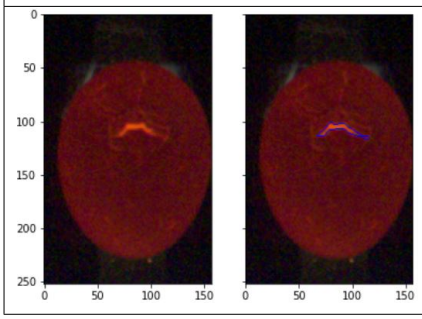
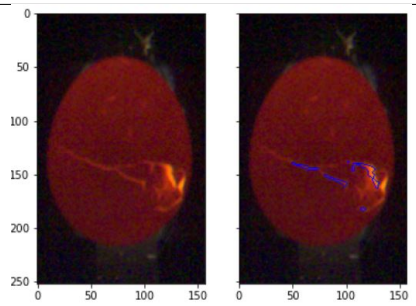
5.1.3 Fissuras

Foram analisadas 872 imagens de ovos brancos e vermelhos, dos quais 347 estavam fissurados e 525 sem fissuras. Uma pequena amostra das imagens e seus resultados é apresentada nos Quadros 12 e 13.

Quadro 12 – Resultados na detecção e classificação de fissuras com algoritmos clássicos em ovos brancos.

Imagem/Detecção	Classificação	Imagem/Detecção	Classificação
	Fissurado		Fissurado
	Sem fissura		Fissurado
	Fissurado		Fissurado
	Fissurado		Sem fissura

Quadro 13 – Resultados na detecção e classificação de fissuras com algoritmos clássicos em ovos vermelhos.

Imagem/Detecção	Classificação	Imagem/Detecção	Classificação
	Fissurado		Sem fissura
	Fissurado		Fissurado
	Fissurado		Fissurado
	Fissurado		Fissurado

A Tabela 13 apresenta a matriz de confusão obtida na classificação das fissuras para ovos vermelhos e para ovos brancos. A partir dessa matriz, as métricas de acurácia, precisão, *Recall* e *F1-score* são obtidas para cada algoritmo (Tabela 14). Como resultado, obteve-se uma precisão média de 99,35 % na classificação de ovos fissurados e uma precisão média de 91,35 % na classificação de ovos sem fissuras. Na métrica *Recall*, o algoritmo obteve uma média de 85,91 % e 99,62 % na classificação de ovos fissurados e sem fissuras, respetivamente. A consequência de haver falsos negativos em ovos brancos e vermelhos, deve-se ao fato de existirem fissuras muito finas para serem classificadas como fissuras (áreas $\leq$ 15), um exemplo desses casos

é apresentado na Figura 102. Outro fator que gerou falsos negativos foi quando o ovo apresentava fissuras muito próximas da borda (Figura 103), pois ao retirar a borda também eram eliminadas as fissuras muito próximas a ele. No entanto, esse pode não ser um problema como no caso de sujeira, onde graças à rotação do ovo seria possível identificar a fissura em outra captura do ovo.

Tabela 13 – Matriz de confusão obtida na classificação de fissuras.

		Vermelhos				**Brancos**	
		Predição				**Predição**	
		Fissurados	**Sem fissura**			**Fissurados**	**Sem fissura**
Real	**Fissurados**	154	38	**Real**	**Fissurados**	142	13
	Sem fissura	2	267		**Sem fissura**	0	256

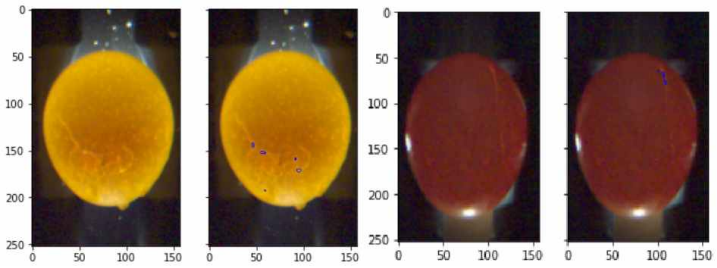
Fonte – O autor.

Tabela 14 – Métricas de avaliação na classificação de fissuras com algoritmos clássicos.

Métricas	**Fissurados**		**Sem fissura**	
	Vermelhos	**Brancos**	**Vermelhos**	**Brancos**
Precisão	0,987	1	0,875	0,952
Recall	0,802	0,916	0,993	1
F1-score	0,885	0,956	0,930	0,975

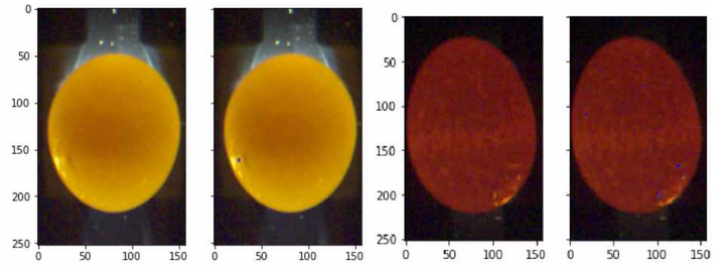
Fonte – O autor.

Figura 102 – Fissuras muito finas que não são identificadas pelo algoritmo de classificação de fissuras.



Fonte – O autor.

Figura 103 – Fissuras muito próximas ao contorno que podem não ser identificadas.



Fonte – O autor.

Um desafio presente nos algoritmos de detecção de fissuras foi a presença de manchas escuras na casca do ovo, essas manchas são apresentadas pela deficiência de cálcio, proteína da matriz e proteína da membrana da casca. Isso não afeta a qualidade interna do ovo, mas reduz a solidez da casca do ovo (ZHANG *et al.*, 2016). As manchas escuras têm formas irregulares, como pontas, listras, flocos, etc. Eles variam em tamanho e número que pode ser de centenas a milhares. Muitas dessas manchas podem ser confundidos com fissuras, pois sua intensidade e forma são muito semelhantes.

Os algoritmos de detecção de fissuras se concentraram em gerar um equilíbrio entre a detecção de rachaduras e a redução do ruído causado por manchas escuras. Dar um peso maior ou menor a um desses fatores depende de se se quer ter menos falsos positivos ou menos falsos negativos. Caso se deseje menor número de falsos negativos, ou seja, garantir que todos os ovos fissurados sejam descartados da linha, deve-se dar um peso maior à detecção de fissuras, aumentando a sensibilidade na classificação, seja diminuindo o valor dos limiares ou diminuindo o tamanho dos filtros. Por outro lado, se se deseja um número menor de falsos positivos, ou seja, que ovos bons não sejam descartados da linha de produção gerando desperdício de produto, deve-se dar um peso maior à redução de ruído, evitando que pequenos ruídos sejam classificados como fissuras. No entanto, encontrar um equilíbrio entre essas duas abordagens é difícil quando se trabalha com técnicas que utilizam limiares para classificação, pois quando o ruído é diminuído, o objeto de interesse também pode estar diminuindo. Por este motivo, na classificação com limiares é difícil obter resultados perfeitos, onde na maioria das vezes alguns fatores devem prevalecer sobre outros.

5.2 RESULTADOS DA CLASSIFICAÇÃO COM ALGORITMOS DE APRENDIZAGEM PROFUNDA

Esta seção descreve os resultados obtidos na avaliação dos dois algoritmos propostos que utilizam aprendizado profundo para a classificação dos ovos: normais, sujos, deformados e fissurados.

5.2.1 Resultados na classificação de imagens

Os modelos *ResNet18*, *34*, *50* e *GoogLeNet* foram avaliados em um conjunto de teste contendo 1.840 imagens. A matriz de confusão dos modelos é apresentada nas Tabelas 26, 27, 28 e 29 respectivamente no apêndice E. As Tabelas 15, 16 e 17 comparam os resultados obtidos na classificação do grau de sujeira, forma do ovo e fissuras nas métricas de precisão (P), *Recall (R)* e *F1-score (F1)*. Para determinar o modelo com um melhor equilíbrio entre precisão e *Recall*, a métrica de *F1-score* é analisada. Da Tabela 15 é possível ver que o modelo *ResNet50* obteve em média

uma percentagem mais elevada de 88,7 % na classificação de ovos com grau de sujeira, superando os modelos *ResNet18*, *34* e *GoogLeNet* em 13 %, 6 % e 10,7 % respectivamente, e uma média de 96,5 % na classificação de ovos limpos. Resultado semelhante foi obtido na classificação do formato do ovo (Tabela 16) onde o *ResNet50* obteve melhor desempenho com um *F1-score* de 84,4 %, superando os modelos *ResNet18*, 3*4* e *GoogLeNet* em 27,73 %, 7,8 % e 16,71 %, respectivamente. No caso da classificação de fissuras (Tabela 17), o modelo *ResNet34* obteve melhor desempenho com um *F1-score* de 94,9 % na classificação de ovos fissuras e um *F1-score* de 93,7 % na classificação de ovos sem fissuras.

Tabela 15 – Resultados obtidos na classificação de imagem no grau de sujeira.

	Grau de sujeira											
	Pouco sujos			**Muito sujos**			**Limpos**			**Média**		
Métricas / Modelos	***P***	***R***	***F1***	***P***	***R***	***F1***	***P***	***R***	***F1***	***P***	***R***	***F1***
ResNet18	0,792	0,730	0,760	0,862	0,690	0,766	0,905	0,857	0,880	0,853	0,759	0,802
ResNet34	0,846	0,813	0,829	0,821	0,855	0,838	0,925	0,900	0,913	0,864	0,856	0,860
ResNet50	0,942	0,854	**0,896**	0,842	0,917	**0,878**	0,968	0,963	**0,965**	0,917	0,911	**0,913**
GoogLeNet	0,868	0,645	0,740	0,832	0,855	0,844	0,849	0,942	0,893	0,850	0,814	0,826

Fonte – O autor.

Tabela 16 – Resultados obtidos na classificação de imagem no formato do ovo.

	Formato do ovo														
	cu			**cd**			**ct**			**cq**			**Média**		
Métricas / Modelos	***P***	***R***	***F1***	***P***	***R***	***F1***	***P***	***R***	***F1***	***P***	***R***	***F1***	***P***	***R***	***F1***
ResNet18	0,816	0,620	0,705	0,928	0,977	0,952	0,769	0,133	0,227	0,765	0,438	0,557	0,819	0,542	0,610
ResNet34	0,780	0,780	0,780	0,956	0,978	0,967	0,824	0,373	0,514	0,910	0,798	0,850	0,868	0,732	0,778
ResNet50	0,860	0,860	**0,860**	0,972	0,985	**0,979**	0,927	0,507	**0,655**	0,916	0,854	**0,884**	0,919	0,801	**0,844**
GoogLeNet	0,897	0,700	0,787	0,934	0,995	0,963	0,867	0,173	0,289	0,909	0,674	0,774	0,902	0,636	0,703

Fonte – O autor.

Tabela 17 – Resultados obtidos na classificação de imagem nas fissuras do ovo.

	Fissuras								
	Fissurados			**Sem fissura**			**Média**		
Métricas / Modelos	***P***	***R***	***F1***	***P***	***R***	***F1***	***P***	***R***	***F1***
ResNet18	0,874	0,964	0,916	0,912	0,626	0,743	0,893	0,795	0,830
ResNet34	1,000	0,903	**0,949**	0,891	0,988	**0,937**	0,946	0,945	**0,943**
ResNet50	1,000	0,873	0,932	0,828	0,988	0,901	0,914	0,930	0,917
GoogLeNet	0,902	0,952	0,926	0,913	0,759	0,829	0,908	0,855	0,878

Fonte – O autor.

Outras categorias avaliadas neste algoritmo foram cor do ovo e imagens sem ovo. Na classificação da cor, os quatro modelos obtiveram um *F1-score* maior que 98 % e na classificação das imagens sem ovos um *F1-score* de 100 %.

A Tabela 18 compara o total de falsos negativos e positivos (*FN* e *FP*) obtidos em cada classe, onde pode-se verificar que os quatro modelos apresentaram falha na classificação das categorias *ct* e *cq* do formato do ovo devido ao alto número de falsos negativos. Essas categorias foram classificadas incorretamente como *cd*, aumentando consideravelmente o número de falsos positivos nessa classe. Isso se deve ao fato de que muito poucas imagens de referência (149 e 331) estavam disponíveis para as classes 3 e 4, em relação as classes 1 e 2 (1.465 e 17.420) veja a Tabela 5. Além disso, podem ser facilmente confundidos por sua semelhança com as categorias vizinhas.

Tabela 18 – Número de falsos positivos e negativos por classe obtidos por cada modelo na classificação de imagens.

		cu	**cd**	**ct**	**cq**	**b_LS**	**v_LS**	**b_LI**	**v_LI**	**ps**	**ms**	**l**	**f**	**sf**	**non**	**Total**
FN	***ResNet18***	19	23	65	50	0	1	1	0	124	45	89	6	31	0	587
	ResNet34	11	22	47	18	0	0	0	0	86	21	62	16	1	0	495
	ResNet50	7	15	37	13	0	0	1	0	67	12	23	21	1	0	343
	GoogLeNet	15	5	62	29	0	1	6	0	163	21	36	8	20	0	484
	Total	52	65	**211**	**110**	0	2	8	0	**440**	**99**	**210**	51	53	0	
FP	***ResNet18***	7	76	3	12	1	0	0	1	88	16	56	23	5	0	421
	ResNet34	11	45	6	7	0	0	0	0	68	27	45	0	10	0	352
	ResNet50	7	28	3	7	0	0	0	1	24	25	20	0	17	0	242
	GoogLeNet	4	71	2	6	1	0	0	5	45	25	104	17	6	0	307
	Total	29	**220**	14	32	2	0	0	7	**225**	**93**	**225**	40	38	0	

Notas: LS: Luz superior
LI: Luz inferior

Fonte – O autor.

Outra falha foi observada na classificação do grau de sujeira, onde os modelos tiveram dificuldade em classificar os ovos com grau intermediário de sujeira (*ps*), alguns foram classificados como limpos, muito sujos ou não foram detectados, aumentando consideravelmente o número de falsos positivos e negativos nessas categorias. Essa lacuna é compreensível pelo fato de não haver fronteiras claras na classificação do grau de sujeira, como é o caso dos ovos limpos, é muito difícil um ovo limpo ser classificado como muito sujo porque existe uma diferença perceptível nestas duas categorias, o que pode ser confirmado nas matrizes de confusão dos modelos. Criar uma fronteira entre um ovo limpo, pouco sujo e muito sujo foi um desafio neste projeto, pois na hora de classificar as imagens para treinamento e teste, havia ovos classificados como limpos alguns dos quais apresentavam um leve grau de sujeira, mas não o suficiente para serem classificados como pouco sujos, da mesma forma que havia ovos que não podiam ser classificados como muito sujos, mas apresentavam alto grau de sujeira, por isso foi difícil classificar as imagens considerando apenas a visão humana como referência.

O modelo *ResNet18* foi o que apresentou maior número de falsos negativos e positivos, principalmente associado a classes não detectadas, também foi insuficiente para classificar ovos sem fissuras, com um total de 31 falsos negativos que diminuíram

seu *Recall* em 62,65 %. Esse modelo também apresentou resultados insuficientes na classificação da categoria *cq*, obtendo um *Recall* de 43,8 %, devido ao elevado número de erros com a categoria *cd*.

O modelo *ResNet50* revelou-se um dos melhores modelos a serem implementados neste processo com uma precisão média de 93,7 %, um *Recall* de 91 % e um *F1-score* de 91,8 % na classificação do ovo por: cor, grau de sujeira, formato e existência de fissuras.

5.2.2 Resultados da segmentação semântica

Os modelos foram avaliados em um conjunto de teste que continha 550 imagens com suas respectivas máscaras. Os resultados são apresentados na Tabela 19. O modelo *Unet-ResNet18* teve melhor desempenho na segmentação do fundo, ovo vermelho, ovo branco e sujeira com um *IoU* de 0,97, 0,95, 0,91 e 0,5, respectivamente. No caso de fissuras, o *Unet-GoogLeNet* foi ligeiramente superior aos outros modelos com um *IoU* de 0,42, superando o ResNet18 em 0,1 %, o *ResNet34* em 2,4 % e o *ResNet50* em 3,9 %. Embora o *IoU* dos defeitos de sujeira e fissuras possam parecer baixos em comparação com as outras classes, isso não significa que os modelos não conseguiram segmentar esses tipos de defeitos. Isso significa que em relação à máscara que contém o rótulo de verdade a previsão não foi exatamente a mesma, uma vez que os defeitos rotulados manualmente não são 100 % precisos. Por esse motivo, não se espera que a previsão seja exatamente igual à máscara. Na Figura 104 pode-se verificar esta afirmação, onde os modelos conseguiram segmentar os defeitos mesmo sem serem totalmente iguais ao rótulo de verdade.

Tabela 19 – Resultados segmentação semântica.

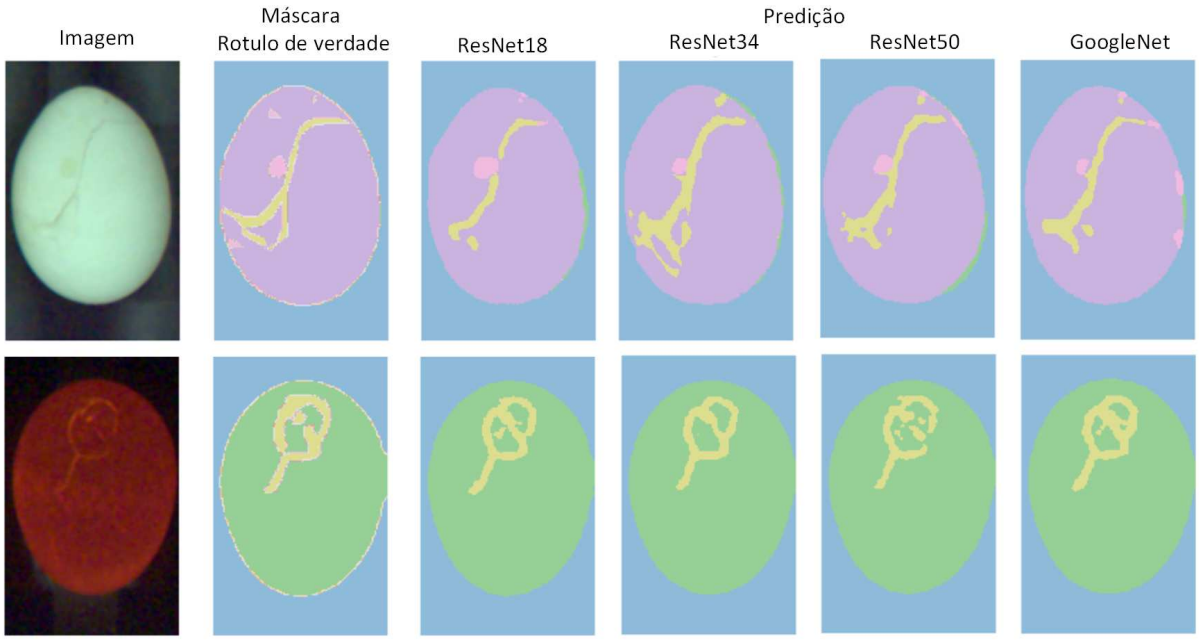
Modelo	***IoU* por classe**					***Iou* Macro**
	Fundo	**Ovo vermelho**	**Ovo branco**	**Sujeira**	**Fissuras**	
Unet-ResNet18	0,9707	0,9557	0,9164	0,5005	0,4263	0,7539
Unet-ResNet34	0,9708	0,9575	0,9125	0,4855	0,4167	0,7486
Unet-ResNet50	0,9698	0,9557	0,9104	0,4927	0,4101	0,7477
Unet-GoogLeNet	0,9706	0,9556	0,9154	0,4780	0,4270	0,7493

Fonte – O autor.

Para avaliar o desempenho dos modelos na classificação, 1.840 imagens de teste foram avaliadas. A matriz de confusão para cada modelo é apresentada nas Tabelas 30, 31, 32 e 33 do apêndice E. As Tabelas 20, 21 e 22 comparam os resultados obtidos na classificação do grau de sujeira, forma do ovo e fissuras a partir das métricas de precisão (P), *Recall* (R) e *F1-score* (F1).

A Tabela 20 mostra que o modelo que obteve o melhor desempenho foi o *Unet-ResNet34* com um *F1-score* médio de 59,5 % e 87,1 % na classificação de ovos sujos

Figura 104 – Amostra de imagens obtidas na segmentação semântica dos quatro modelos avaliados.



Fonte – O autor.

e limpos, respectivamente. Resultado semelhante foi observado na classificação da forma do ovo (Tabela 21) e na classificação das fissuras (Tabela 22), onde o *Unet-ResNet34* obteve um desempenho superior com um *F1-score* de 67 %, 99,7 % e 96,93 % na classificação do formato do ovo, ovos fissurados e ovos sem fissuras, respectivamente. Em relação à classificação da cor do ovo e imagens sem ovos, os quatro modelos obtiveram uma precisão de 100 %.

Tabela 20 – Resultados obtidos na classificação do grado de sujeira usando segmentação semântica.

	Grau de sujeira											
	Pouco sujos			**Muito sujos**			**Limpos**			**Média**		
Métricas / **Modelos**	***P***	***R***	***F1***	***P***	***R***	***F1***	***P***	***R***	***F1***	***P***	***R***	***F1***
Unet-ResNet18	0,659	0,563	0,607	0,461	0,697	0,555	0,838	0,833	0,836	0,653	0,697	0,666
Unet-ResNet34	0,738	0,536	**0,621**	0,479	0,703	**0,570**	0,829	0,916	**0,871**	0,682	0,718	**0,687**
Unet-ResNet50	0,712	0,544	0,617	0,460	0,710	0,558	0,826	0,873	0,849	0,666	0,709	0,675
Unet-GoogLeNet	0,653	0,523	0,581	0,401	0,683	0,505	0,848	0,838	0,843	0,634	0,681	0,643

Fonte – O autor.

Tabela 21 – Resultados obtidos na classificação do formato do ovo usando segmentação semântica.

	Formato do ovo														
	cu			**cd**			**ct**			**cq**			**Média**		
Métricas / Modelos	***P***	***R***	***F1***	***P***	***R***	***F1***	***P***	***R***	***F1***	***P***	***R***	***F1***	***P***	***R***	***F1***
Unet-ResNet18	0,429	0,600	0,500	0,971	0,897	0,932	0,534	0,733	0,618	0,441	0,584	0,502	0,594	0,704	0,666
Unet-ResNet34	0,469	0,600	**0,526**	0,964	0,930	**0,947**	0,527	0,773	**0,627**	0,623	0,539	**0,578**	0,646	0,711	**0,687**
Unet-ResNet50	0,469	0,600	0,526	0,967	0,928	0,947	0,500	0,760	0,603	0,539	0,461	0,497	0,619	0,687	0,675
Unet-GoogLeNet	0,508	0,600	0,550	0,971	0,914	0,942	0,588	0,800	0,678	0,455	0,573	0,507	0,631	0,722	0,643

Fonte – O autor.

Tabela 22 – Resultados obtidos na classificação de fissuras usando segmentação semântica.

	Fissuras								
	Fissurados			**Sem fissura**			**Média**		
Métricas / Modelos	***P***	***R***	***F1***	***P***	***R***	***F1***	***P***	***R***	***F1***
Unet-ResNet18	1	0,501	0,667	0,9156	0,915	0,915	0,958	0,708	0,792
Unet-ResNet34	1	0,99	**0,997**	0,9875	0,951	**0,969**	0,994	0,973	**0,983**
Unet-ResNet50	0,982	0,994	0,988	0,988	0,964	0,976	0,985	0,979	0,982
Unet-GoogLeNet	0,976	0,988	0,982	0,975	0,952	0,963	0,976	0,970	0,973

Fonte – O autor.

A Tabela 23 compara o número de falsos positivos e negativos de cada classe obtidos pelo algoritmo de segmentação na classificação do ovo por formato, cor, grau de sujeira, fissuras e imagens sim ovo. Aqui, pode-se observar que o algoritmo proposto de segmentação neste conjunto de dados, não foi possível reconhecer de forma satisfatória o formato do ovo, uma vez que este apresentou um grande número de falsos negativos e positivos entre as quatro formas do ovo.

Tabela 23 – Número de falsos positivos e negativos por classe obtidos por cada modelo na classificação usando segmentação semântica.

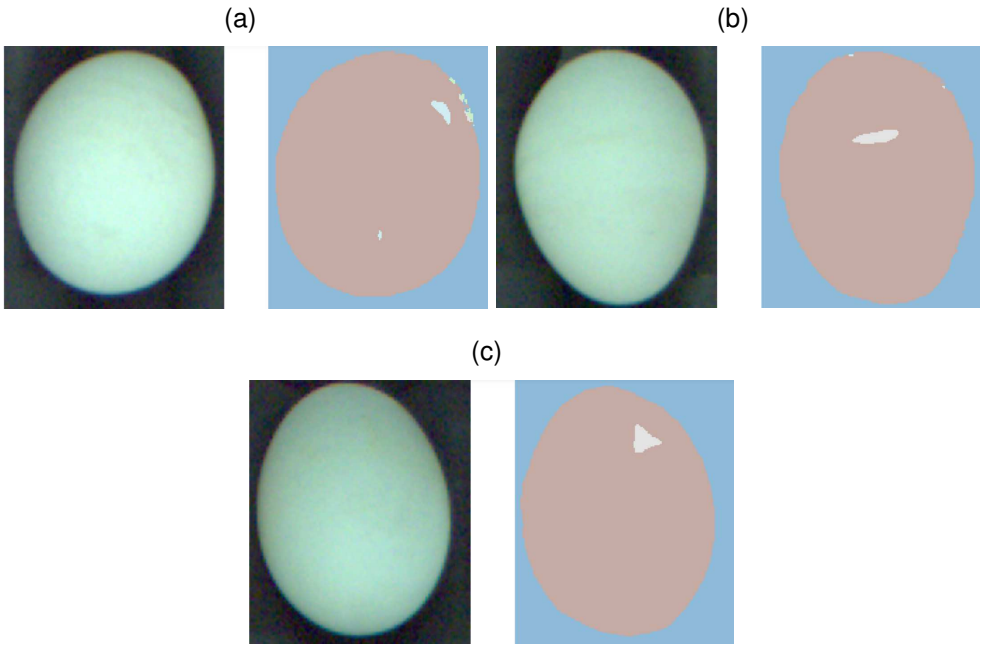
		cu	**cd**	**ct**	**cq**	**b_LS**	**v_LS**	**b_LI**	**v_LI**	**ps**	**ms**	**l**	**f**	**sf**	**non**	**Total**
FN	***ResNet18***	20	104	20	37	0	0	0	0	209	44	100	7	7	0	548
	ResNet34	20	71	17	42	0	0	0	0	222	43	50	1	4	0	470
	ResNet50	20	72	18	48	0	0	0	0	218	42	76	1	3	0	498
	GoogLeNet	20	86	15	38	0	0	0	0	228	46	97	2	4	0	536
	Total	**80**	**333**	**70**	**165**	0	0	0	0	**877**	**175**	**323**	11	18	0	
FP	***ResNet18***	40	27	48	66	0	0	0	0	139	118	96	7	7	0	548
	ResNet34	34	35	52	29	0	0	0	0	91	111	113	0	1	0	466
	ResNet50	34	32	57	35	0	0	0	0	105	121	110	3	1	0	498
	GoogLeNet	29	27	42	61	0	0	0	0	133	148	90	4	2	0	536
	Total	**137**	**121**	**199**	**191**	0	0	0	0	**468**	**498**	**409**	14	11	0	

Notas: LS: Luz superior
LI: Luz inferior

Fonte – O autor.

Da mesma forma, o número total de falsos negativos e positivos na classificação do grau de sujeira foi alto, isso se deveu ao fato do modelo de segmentação ser extremamente sensível na detecção de sujeira, onde foram detectadas manchas quase imperceptíveis à visão humana. Um exemplo disso é apresentado na Figura 105. Essa alta sensibilidade fez com que os ovos que inicialmente foram considerados em uma categoria, sejam limpos, pouco sujos ou muito sujos, fossem classificados em outra. Isso explicaria o alto número de falsos positivos e negativos obtidos. Vale ressaltar o resultado desse algoritmo na classificação de fissuras, em geral os modelos obtiveram um baixo número de falsos negativos e positivos que permitiram obter índices de 99,4 % e 97,3 % na precisão e *Recall* respectivamente.

Figura 105 – Amostras de detecção de sujeira usando segmentação semântica.



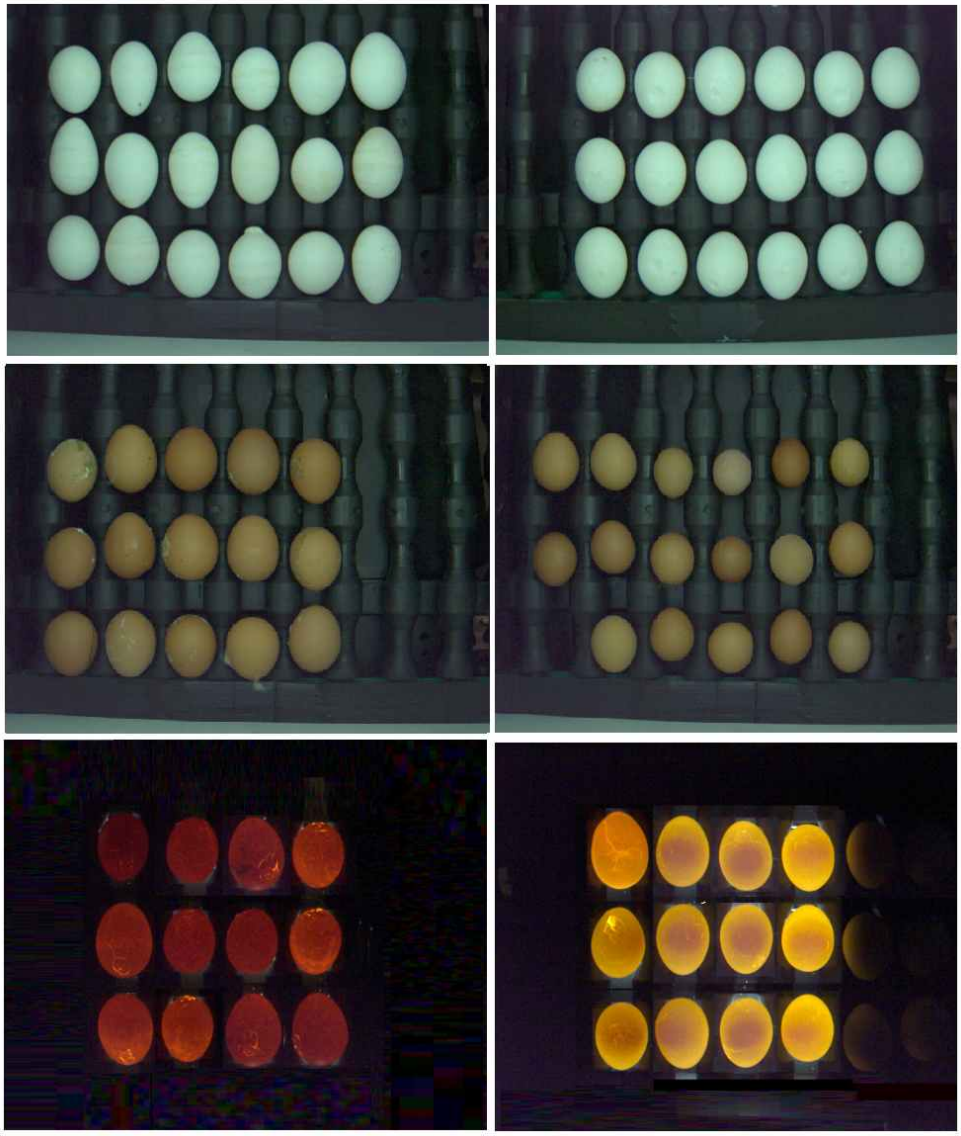
Fonte – O autor.

Nesse algoritmo, o modelo com melhor desempenho foi o *Unet-ResNet34* com uma precisão média de 83 %, um *Recall* de 85,1 % e um *F1-score* de 83,5 % na classificação do ovo por: cor, grau de sujeira, forma e existência de fissuras.

5.3 COMPARAÇÃO DOS ALGORITMOS CLÁSSICOS E DE APRENDIZAGEM PROFUNDA

Um total de 70 imagens foram selecionadas para avaliação, um exemplo destas é apresentado na Figura 106. A proporção do tamanho máximo de sujeira permitido foi definida no valor mínimo de 0,0104, isso implica que um ovo será classificado como sujo com qualquer tamanho de mancha detectado, portanto, se a previsão nos algoritmos for *ps* ou *ms*, o ovo será classificado em uma única categoria como sujo (*s*).

Figura 106 – Imagens de teste para comparação dos algoritmos.



Fonte – O autor.

A forma de ovo definida para rejeição foi a forma 10 (*cq*). O tempo de início e término do processamento de cada ovo foi medido desde o momento em que a imagem entra no algoritmo até a entrega da previsão. Os testes para medir o tempo de processamento utilizaram um computador CPU Intel Core i5-9300H 2,4 GHz com 4 núcleos; GPU NVIDIA GeForce GTX 1060 Intel UHD Graphics 630 e 4 GB de RAM.

A Tabela 24 mostra a comparação dos algoritmos com as métricas de acurácia, precisão, *Recall*, *F1-score* e tempo de processamento. Para determinar o algoritmo com melhor desempenho, foi utilizado um método conhecido como *Multi Criteria Decision Making (MCDM)* (TRIANTAPHYLLOU, 2000). Este método permite escolher a melhor alternativa com base nos objetivos e preferências do projeto, neste caso se procura encontrar um algoritmo com um melhor equilíbrio entre as métricas e o tempo de processamento, por isso foi atribuído um peso de 0,2 aos cinco critérios de avaliação

para tomada de decisão. Esses pesos permitiram obter o vetor de prioridade de cada algoritmo, onde se pode observar que o algoritmo que apresentou melhor desempenho nessas condições foi o algoritmo clássico, que foi o algoritmo com menor tempo de processamento de 0,049 ms por cada imagem do ovo, uma acurácia de 81,07 % e um *F1-score* de 75,78 % na classificação da cor do ovo, o formato do ovo, o grau de sujeira e as fissuras. O segundo algoritmo foi o de classificação de imagens com a arquitetura *ResNet18* com uma acurácia de 86,07 %, um *F1-score* de 71,97 % e um tempo de processamento de 0,074 ms.

O algoritmo que obteve maior acurácia foi o algoritmo de segmentação semântica com a arquitetura *ResNet34* com acurácia de 88,92 % na classificação e um tempo de processamento de 0,47 ms. No caso da arquitetura *GoogLeNet*, esperava-se que fosse mais rápida que as arquiteturas *ResNet18* e *ResNet34* devido ao fato de possuir menos operações, porém, *GoogLeNet* obteve um tempo maior de 0,041 e 0,00245 ms respectivamente no algoritmo de classificação de imagens e 1,209 e 1,181 ms maior na segmentação semântica.

Tabela 24 – Comparação dos algoritmos usando *MCDM*.

Algoritmo		**Tempo Médio (ms)**	**Acurácia**	**Precisão**	**Recall**	**F1-score**	**Análise multicritério**
Clássico		0,0497	0,811	0,772	0,757	0,758	**0,9148**
Classificação de imagens	*ResNet18*	0,0749	0,861	0,732	0,726	0,719	0,8331
	ResNet34	0,1135	0,850	0,732	0,877	0,734	0,8234
	ResNet50	0,1722	0,875	0,801	0,872	0,776	0,8239
	GoogLeNet	0,1159	0,850	0,741	0,863	0,743	0,8227
Segmentação semântica	*ResNet18*	0,4435	0,875	0,834	0,845	0,838	0,8053
	ResNet34	0,4709	0,889	0,701	0,859	0,712	0,7493
	ResNet50	2,8645	0,868	0,809	0,857	0,828	0,7790
	GoogLeNet	1,6527	0,883	0,862	0,844	0,839	0,7969

Fonte – O autor.

A fim de definir os melhores algoritmos para cada classificação individualmente, é apresentada a Tabela 25 onde é feita a comparação das métricas de precisão (*P*), *Recall* (*R*) e *F1-score* (*F1*) obtidas por cada modelo na classificação da sujeira, formato, fissuras e cor do ovo. Na classificação da sujeira é necessário analisar a métrica *Recall*, pois, como já foi mencionado, é fundamental garantir o menor número de falsos negativos, ou seja, o menor número de ovos sujos categorizados como limpos. Nesta classificação, os modelos obtiveram um *Recall* superior a 84,9 %, sendo o algoritmo de segmentação semântica com a arquitetura *ResNet34* o que obteve o melhor desempenho com um *Recall* de 96,6 %.

Na classificação do formato do ovo, a métrica de *F1-score* é analisada porque, espera-se ter o menor número de falsos negativos e o menor número de falsos positivos, neste caso o algoritmo com melhor desempenho foi o algoritmo de segmentação

semântica com a arquitetura *GoogLeNet* com um *F1-score* de 74 % seguido pelo algoritmo clássico com um F1-score de 72,4 %.

Na classificação de fissuras, a métrica *Recall* é analisada de forma a obter o modelo com menor número de falsos negativos, neste caso os modelos de segmentação semântica com as arquiteturas *ResNet18* e *50* apresentaram melhor desempenho com um *Recall* de 94,1 %. Finalmente, na detecção da cor, a maioria dos modelos obtiveram resultados em *F1-score* superiores a 99 %, com exceção do algoritmo clássico que apresentou 11 falsos positivos na classificação do ovo branco, isso ocorreu porque existem ovos vermelhos que apresentam uma ligeira mudança de matiz na casca muito próxima ao branco onde o algoritmo não foi capaz de diferenciar esses tons.

Tabela 25 – Comparação das métricas obtidas por cada modelo na classificação da sujeira, formato, fissuras e cor do ovo.

Algoritmo		Sujeira			Formato			Fissuras			Cor		
		P	***R***	***F1***	***P***	***R***	***F1***	***P***	***R***	***F1***	***P***	***R***	***F1***
Clássico		0,720	**0,916**	0,852	0,751	0,706	**0,724**	0,688	**0,704**	0,682	0,688	0,704	**0,682**
Classificação de imagens	*ResNet18*	0,906	**0,933**	0,953	0,576	0,533	**0,550**	0,621	**0,632**	0,597	0,995	0,996	**0,995**
	ResNet34	0,917	**0,916**	0,952	0,546	0,787	**0,542**	0,739	**0,787**	0,670	1	0,996	**0,998**
	ResNet50	0,919	**0,891**	0,938	0,594	0,783	**0,557**	0,738	**0,795**	0,662	1	1	**1**
	GoogLeNet	0,869	**0,849**	0,902	0,614	0,868	**0,604**	0,656	**0,678**	0,642	0,995	0,996	**0,995**
Segmentação semântica	*ResNet18*	0,907	**0,924**	0,940	0,678	0,701	**0,685**	0,905	**0,941**	0,894	1	1	**1**
	ResNet34	0,938	**0,966**	0,962	0,447	0,704	**0,448**	0,781	**0,839**	0,725	1	1	**1**
	ResNet50	0,898	**0,933**	0,937	0,658	0,697	**0,673**	0,905	**0,941**	0,893	1	1	**1**
	GoogLeNet	0,888	**0,908**	0,927	0,721	0,787	**0,740**	0,855	**0,789**	0,827	1	1	**1**

Fonte – O autor.

A partir dos resultados obtidos, foi possível evidenciar que, neste tipo de processo, os métodos de segmentação semântica são ferramentas promissoras para a identificação de defeitos, principalmente o melhor desempenho foi observado com a arquitetura *unet-ResNet* na detecção de manchas de sujeira e fissuras em os ovos, com taxas médias de precisão de 91,4 % e 86,39 % e um *Recall* de 94,12 % e 90,69 % respectivamente. Não entanto, foram os métodos que utilizaram um tempo de processamento maior (> 0,4 ms) em relação ao algoritmo clássico e ao algoritmo de classificação de imagens. Isto é porque estes modelos empregam um maior numero de operações, sua arquitetura é mais complexa, e além disso por que foram empregados algoritmos adicionais para realizar a classificação como o algoritmo de formato.

Por outro lado, os algoritmos de classificação de imagens com CNN obtiveram resultados intermediários entre métricas de avaliação e tempo de processamento. As falhas encontradas nestes algoritmos descritos na seção 5.2 são falhas que podem ser melhoradas aumentando o número de exemplos de cada classe. Também é aconselhável usar algoritmos de suporte computacional na criação do banco de dados para o treinamento dos modelos. Por exemplo, no caso do grau de sujeira, esta classificação foi extenuante e susceptível a erro humano. Se for utilizado um algoritmo que possa

calcular a porcentagem de sujeira em relação à área total do ovo, é possível usar esses cálculos para fazer uma classificação com limites mais visíveis, o que poderia melhorar significativamente as métricas de avaliação.

No caso dos algoritmos clássicos, estes provaram ser algoritmos com um tempo de resposta rápido (< 0,04 ms) e com uma acurácia de 81,01 % na classificação de sujeira, fissuras e formato. Foi possível garantir esse desempenho nos algoritmos porque foram executados em um ambiente controlado onde parâmetros como iluminação, escala e rotação nas imagens não apresentavam grandes variações. No caso de apresentar variações nas imagens, os algoritmos clássicos diminuem seu desempenho, pois são dependentes de limites e máscaras definidos com base em parâmetros específicos das imagens no momento da criação dos algoritmos.

Uma vantagem das abordagens clássicas e as baseadas na segmentação semântica neste trabalho, foi que permitiram atender aos requisitos funcionais do projeto de forma mais dinâmica em relação às técnicas de classificação de imagens com CNN. Isso porque, por exemplo, na classificação de sujeira em ovos, um dos requisitos é que seja possível determinar o grau de mancha que o usuário define para enviar para o processo de lavagem, com técnicas clássicas e de segmentação essa configuração é definida a partir de um limiar que o usuário pode alterar a qualquer momento. No caso das técnicas de classificação de imagens com CNN, essa configuração não era tão intuitiva, pois para classificar diferentes graus de sujeira era necessário treinar os modelos com categorias que consideravam diferentes tipos de sujeira, o que não foi um processo simples. Outro exemplo é apresentado na classificação da forma do ovo, esta classificação obteve um melhor desempenho na métrica *F1-score* com as técnicas clássicas e um resultado não superior a 60 % com o algoritmo de classificação de imagens, conforme observado nos resultados da Tabela 25. Isso se deve ao fato de que nas técnicas clássicas são utilizados parâmetros numéricos para diferenciar as classes, esses parâmetros permitem obter fronteiras mais visíveis entre as categorias, principalmente para formatos muito semelhantes, facilitando sua classificação.

5.4 CONSIDERAÇÕES FINAIS

Após a realização dos experimentos, é possível concluir que as abordagens clássicas com o hardware utilizado neste trabalho apresentaram um melhor equilíbrio em termos de tempo de resposta e precisão. Essas abordagens permitiram detectar várias características dos ovos como cor, formato e defeitos na superfície da casca do ovo, com um tempo de resposta relativamente rápido em comparação com os outros modelos.

De acordo com a literatura, as abordagens clássicas podem se tornar complexas porque requerem vários níveis de processamento. Esse questionamento foi confirmado, essas abordagens apresentaram algumas dificuldades na sua implementação, vários

testes tiveram que ser realizados com diferentes algoritmos para determinar quais abordagens permitiam classificar as características do ovo de forma satisfatória e computacionalmente acessível. Realizar a integração de todos os algoritmos desde a segmentação, aplicação de filtros, extração de características até a tomada de decisão, exigiu um tempo de implementação estendido e conhecimento avançado na aplicação de técnicas de processamento de imagens. Por outro lado, as abordagens baseadas em aprendizagem profundo não requerem esta série de implementações uma vez que essas abordagens recebem uma imagem com seu respectivo rótulo e a rede realiza todo o trabalho de extração das características obtendo um modelo capaz de decidir sobre novas imagens de entrada. Isso elimina todo o fluxo de processamento envolto em técnicas clássicas facilitando sua implementação nesse sentido.

Outra descoberta importante nesta pesquisa foi que os métodos propostos com abordagens clássicas e baseadas em aprendizagem podem ser implementados no sistema de inspeção de ovos desenvolvido para este projeto, com exceção do método de segmentação semântica que usa o *unet-ResNet50*. Conforme definido em um dos requisitos não funcionais do trabalho, este sistema deve operar a uma frequência de 60 Hz. A partir das configurações de captura, duas imagens com 18 ovos cada devem ser processadas em um intervalo de 0-92 ms. Nesse caso, todos os métodos propostos, em média, usaram menos de 17 ms para fazer essas previsões, exceto a arquitetura *unet-ResNet50*, que, em média, exigiu 103 ms. Portanto, pode-se afirmar que, neste trabalho, nem todos os algoritmos baseados em aprendizagem profunda puderam ser implementados no sistema de inspeção de ovos em tempo real. A implementação destes algoritmos depende de fatores como o método escolhido, o número de operações utilizadas, a complexidade da arquitetura implementada, o custo computacional envolvido nos algoritmos adicionais e o hardware associado à captura das imagens e ao processamento dos algoritmos.

6 CONCLUSÃO

Este trabalho realiza uma comparação de três métodos de processamento de imagens aplicados a um problema de classificação de ovos de galinha em quatro categorias: normal, sujo, geometricamente anormal e quebrado. O primeiro método usa técnicas clássicas de processamento de imagem, como binarização, detecção de contorno, remoção de ruído, contagem de pixels e seleção de limites para tomada de decisão. O segundo e o terceiro métodos usam redes neurais convolucionais para a classificação. Uma série de experimentos foi realizada para medir quantitativamente o desempenho de cada método.

Nesta pesquisa foi possível comprovar que as técnicas de visão computacional podem ser utilizadas em tarefas de inspeção e classificação. Essas técnicas podem resolver os problemas associados às tarefas de inspeção realizadas manualmente, evitando danos à saúde dos trabalhadores. Os algoritmos desenvolvidos neste trabalho possibilitaram detectar e classificar os ovos com uma acurácia superior a 80 %.

Um dos desafios deste trabalho foi a seleção do equipamento para o sistema de inspeção. Isso se deve ao fato de a maioria dos trabalhos encontrados na literatura terem sido elaborados para testes de laboratório e não para operação industrial. Vários testes foram necessários para definir o sistema de aquisição, como iluminação, câmeras e o compartimento fechado para a captura das imagens. A desvantagem do protótipo implementado é que não é possível detectar defeitos internos nos ovos, como manchas de sangue ou carne. Isto porque as lâmpadas de iluminação inferior não puderam ser localizadas próximas à superfície do ovo devido à estrutura física do módulo de movimentação, mais especificamente por causa ao desenho dos rolos sobre os quais os ovos eram apoiados. A luz da lâmpada deve estar localizada o mais próximo possível do ovo evitando vazamentos de luz para realizar a análise do interior da casca.

A partir dos experimentos, constatou-se que as abordagens clássicas são ferramentas muito úteis para o campo da visão computacional. Nesse trabalho, provaram ser alternativas rápidas e acessíveis. No entanto, é necessário considerar que o desempenho destas técnicas é fortemente dependente da correta seleção dos algoritmos a serem implementados. A realização de uma análise de custo-benefício é essencial para obter um equilíbrio entre a precisão necessária e os recursos computacionais disponíveis. As técnicas clássicas utilizadas para o desenvolvimento dos algoritmos de classificação neste trabalho obtiveram um desempenho ligeiramente superior às técnicas de aprendizagem profunda, considerando o tempo de processamento e o desempenho nas métricas de avaliação.

As abordagens baseadas na aprendizagem profunda do ponto de vista operacional, são abordagens simples onde a dinâmica é a mesma, uma imagem é introduzida

como entrada e uma saída é obtida como resposta. No entanto, pode haver fatores que dificultam sua implementação, como por exemplo, é necessário garantir um número considerável de exemplos das categorias a serem classificadas. Essa não é uma tarefa simples, é necessário ter pessoal com conhecimento na detecção de defeitos em ovos de galinha para criar um banco de dados confiável, corretamente rotulado e de tamanho suficiente para garantir um grau eficiente de generalização pelos modelos. Além disso, esta abordagem requer hardware com alto poder de processamento, devido a cálculos pesados, como convoluções, multiplicações de matrizes e funções de ativação. Requer memória suficiente para armazenar todas as informações relacionadas aos parâmetros da rede e uma placa gráfica que garanta uma velocidade de processamento de acordo com os requisitos do projeto.

O sistema de inspeção com os algoritmos propostos atingiu os objetivos do trabalho, resolvendo as questões de pesquisa e atendendo a todos os requisitos funcionais e não funcionais propostos. Os três métodos desenvolvidos para classificação demonstraram ter potencial para serem implementados no produto comercial desenvolvido para a classificação de ovos em indústrias avícolas. Contudo, ainda é necessário aumentar a precisão nos algoritmos.

6.1 TRABALHOS FUTUROS

A partir da análise dos resultados obtidos e das dificuldades observadas durante o projeto, foram propostas algumas tarefas que poderiam ser desenvolvidas em torno deste tema. Para trabalhos futuros, em abordagens de aprendizagem profunda sugere-se experimentar um conjunto de dados maior. Para isso, não só é necessário um maior número de imagens de ovos, mas também é necessária uma maior precisão na rotulagem das imagens. Nesse estágio, é essencial ter rótulos precisos para que a rede neural possa aprender e generalizar para alcançar resultados satisfatórios.

Neste estudo, duas arquiteturas de rede neural foram selecionadas para serem utilizadas nos experimentos. Como critério de seleção, considerou-se que deveriam apresentar um equilíbrio entre uso de memória, precisão e número de operações. Considerando o progresso nesta área de investigação, certamente surgirão modelos que apresentem um melhor equilíbrio entre estes critérios que podem ser testados e avaliados no contexto deste projeto.

Avaliar outros tipos de abordagens de aprendizado profundo, como modelos de detecção de objetos, como *SSD* (LIU *et al.*, 2016) ou modelos da família *YOLO* (REDMON; FARHADI, 2017) para realizar a classificação e detecção de defeitos.

A abordagem clássica obteve melhor pontuação na análise multicritério, onde o tempo de processamento foi determinante nos resultados. Tanto as abordagens clássicas quanto as baseadas em aprendizagem profunda foram executadas em CPU. Sugere-se executar esses algoritmos em uma placa gráfica GPU e realizar uma nova

análise multicritério para determinar qual abordagem tem melhor desempenho nessas condições.

Para a detecção automática de defeitos internos, um problema que ainda não foi resolvido no cenário atual, trabalhos e estudos futuros são necessários para determinar como esses defeitos podem ser detectados com o sistema de inspeção atual ou se modificações devem ser feitas na estrutura para permitem analisar este defeito.

REFERÊNCIAS

ABBASPOUR-GILANDEH, YOUSEF; AZIZI, AFSHIN. Identification of Cracks in Eggs Shell Using Computer Vision and Hough Transform. **Yüzüncü Yıl Üniversitesi Tarım Bilimleri Dergisi**, v. 28, n. 4, p. 375–383, 2018.

ABDULLAH, MH; NASHAT, S; ANWAR, SA; ABDULLAH, MZ. A framework for crack detection of fresh poultry eggs at visible radiation. **Computers and Electronics in Agriculture**, Elsevier, v. 141, p. 81–95, 2017.

AGRONEGÓCIO, FIESP DEPARTAMENTO DE. **OUTLOOK FIESP- Projeções para o agronegócio brasileiro 2028**. Fev. 2020. Disponível em: http://outlookdeagro.azurewebsites.net/OutLookDeagro/pt-BR/Publicacao/Ovos/Brasil/2028.

ALON, Alvin. An Image Processing Approach of Multiple Eggs' Quality Inspection. **International Journal of Advanced Trends in Computer Science and Engineering**, v. 8, p. 2794–2799, dez. 2019. DOI: 10.30534/ijatcse/2019/18862019.

ARIVAZHAGAN, S; SHEBIAH, R Newlin; SUDHARSAN, Hariharan; KANNAN, R Rajesh; RAMESH, R. External and internal defect detection of egg using machine vision. **Journal of Emerging Trends in Computing and Information Sciences**, Citeseer, v. 4, n. 3, p. 257–262, 2013.

BADRINARAYANAN, Vijay; KENDALL, Alex; CIPOLLA, Roberto. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. **IEEE transactions on pattern analysis and machine intelligence**, IEEE, v. 39, n. 12, p. 2481–2495, 2017.

BI, Xin. **Environmental Perception Technology for Unmanned Systems**. [*S.l.*]: Springer.

BIGGINS, John D; THOMPSON, Jamie E; BIRKHEAD, Tim R. Accurately quantifying the shape of birds' eggs. **Ecology and Evolution**, Wiley Online Library, v. 8, n. 19, p. 9728–9738, 2018.

BLESSO, Christopher N. Egg phospholipids and cardiovascular health. **Nutrients**, Multidisciplinary Digital Publishing Institute, v. 7, n. 4, p. 2731–2747, 2015.

BROWNLEE, Jason. **How to use Learning Curves to Diagnose Machine Learning Model Performance**. Fev. 2019. Disponível em: https://machinelearningmastery.com/learning-curves-for-diagnosing-machine-learning-model-performance/. Acesso em: 10 mar. 2020.

CANZIANI, Alfredo; PASZKE, Adam; CULURCIELLO, Eugenio. An analysis of deep neural network models for practical applications. **arXiv preprint arXiv:1605.07678**, 2016.

CDFA. **EGG SAFETY & QUALITY MANAGEMENT PROGRAM**. 2016. Disponível em: https://ucanr.edu/sites/EDC_Master_Gardeners/files/235007.pdf. Acesso em: 9 jan. 2021.

DE KETELAERE, Bart; COUCKE, P; DE BAERDEMAEKER, Josse. Eggshell crack detection based on acoustic resonance frequency analysis. **Journal of Agricultural Engineering Research**, Elsevier, v. 76, n. 2, p. 157–163, 2000.

DEHROUYEH, MH; OMID, M; AHMADI, H; MOHTASEBI, SS; JAMZAD, M. Grading and quality inspection of defected eggs using machine vision. **International Journal of Advanced Science and Technology**, Citeseer, v. 17, n. 4, p. 23–30, 2010.

ECKELS, Joshua. **Multi-class semantic segmentation metrics and accuracy**. [*S.l.*: *s.n.*], set. 2020. Disponível em: https://forums.fast.ai/t/multi-class-semantic-segmentation-metrics-and-accuracy/74665/3. Acesso em: 14 dez. 2020.

ELBY. **Least squares circle**. Mar. 2011. Disponível em: https://scipy-cookbook.readthedocs.io/items/Least_Squares_Circle.html. Acesso em: 20 abr. 2020.

FAROOQ, Muhammad; SAZONOV, Edward. Feature extraction using deep learning for food type recognition. *In*: SPRINGER. INTERNATIONAL conference on bioinformatics and biomedical engineering. [*S.l.*: *s.n.*], 2017. P. 464–472. doi.org/10.1007/978-3-319-56148-$6_4$1.

FASTAI. **Data block**. Set. 2020. Disponível em: https://docs.fast.ai/data.block. Acesso em: 3 set. 2020.

FEI-FEI, Li; RANJAY, Krishna; DANFEI, Xu. **Lecture 9: CNN Architectures**. 136 slides. Mai. 2020. Disponível em:

http://cs231n.stanford.edu/slides/2020/lecture_9.pdf. Acesso em: 6 ago. 2020.

FEYAERTS, Filip; VANROOSE, Peter; FRANSENS, Rik; VAN GOOL, Luc J. Using shape to correct for observed nonuniform color in automated egg grading. *In*: INTERNATIONAL SOCIETY FOR OPTICS e PHOTONICS. MACHINE Vision Applications in Industrial Inspection IX. [*S.l.*: *s.n.*], 2001. P. 93–101.

GARCÍA-ALEGRE, María C; RIBEIRO, Angela; GUINEA, Domingo; CRISTÓBAL, Gabriel. Color index analysis for automatic detection of eggshell defects. *In*: PROC. SPIE. [*S.l.*: *s.n.*], 2000. P. 380–387.

GEISLER, Wilson. Vision: A Computational Investigation into the Human Representation and Processing of Visual Information. **PsycCRITIQUES**, v. 28, n. 8, p. 581–582, 1983.

GIELEN, Robert MAM; DE JONG, Leo P; KERKVLIET, Harry MM. Electrooptical blood-spot detection in intact eggs. **IEEE Transactions on Instrumentation and Measurement**, IEEE, v. 28, n. 3, p. 177–183, 1979.

GILL, Navdeep; HALL, Patrick; MONTGOMERY, Kim; SCHMIDT, Nicholas. A Responsible Machine Learning Workflow with Focus on Interpretable Models, Post-hoc Explanation, and Discrimination Testing. **Information**, Multidisciplinary Digital Publishing Institute, v. 11, n. 3, p. 137, 2020.

GONZALEZ, Rafael C.; WOODS, Richard E. **Digital Image Processing**. second. [*S.l.*]: New Jersey: Pearson Education, Inc., 2004.

GOODRUM, J W; ELSTER, RT. Machine vision for crack detection in rotating eggs. **Transactions of the ASAE**, American Society of Agricultural e Biological Engineers, v. 35, n. 4, p. 1323–1328, 1992.

GUANJUN, Bao; MIMI, Jia; YI, Xun; SHIBO, Cai; QINGHUA, Yang. Cracked egg recognition based on machine vision. **Computers and electronics in agriculture**, Elsevier, v. 158, p. 159–166, 2019.

HAUVER, William Eugene. **Egg grading manual**. [*S.l.*]: US Department of Agriculture, Agricultural Marketing Service, 1961.

HAVLÍČEK, Miroslav; NEDOMOVÁ, Šárka; SIMEONOVOVÁ, Jana; SEVERA, Libor; KŘIVÁNEK, Ivo *et al.* On the evaluation of chicken egg shape variability. **Acta Universitatis Agriculturae et Silviculturae Mendelianae Brunensis**, Mendel University Press, v. 56, n. 5, p. 69–74, 2008.

HE, Kaiming; ZHANG, Xiangyu; REN, Shaoqing; SUN, Jian. 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.

HOWARD, Jeremy. **Image Classification**. Lesson 4 - Deep Learning for Coders (2020) Fastai. 2020a. Disponível em: `https://github.com/fastai/fastbook/blob/master/05_pet_breeds.ipynb`. Acesso em: 9 out. 2020.

HOWARD, Jeremy. **Lesson 1: Image classification**. 1 video (100 min). 2019. Disponível em: `https://course.fast.ai/videos/?lesson=1]`. Acesso em: 15 mar. 2020.

HOWARD, Jeremy. **Lesson 2 - Deep Learning for Coders (2020)**. 1 video (81 min). Ago. 2020b. Disponível em: `https://www.youtube.com/watch?v=BvHmRx14HQ8`. Acesso em: 3 set. 2020.

HOWARD, Jeremy; GUGGER, Sylvain. Fastai: A layered API for deep learning. **Information**, Multidisciplinary Digital Publishing Institute, v. 11, n. 2, p. 108, 2020. doi.org/10.3390/info11020108.

HU, Jie; SHEN, Li; SUN, Gang. Squeeze-and-excitation networks. *In*: PROCEEDINGS of the IEEE conference on computer vision and pattern recognition. [*S.l.*: *s.n.*], 2018. P. 7132–7141.

IOFFE, Sergey; SZEGEDY, Christian. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. **CoRR**, abs/1502.03167, 2015. arXiv: `1502.03167`. Disponível em: `http://arxiv.org/abs/1502.03167`. Acesso em: 10 mar. 2020.

JIN, Cheng; XIE, Lijuan; YING, Yibin. Eggshell crack detection based on the time-domain acoustic signal of rolling eggs on a step-plate. **Journal of Food Engineering**, Elsevier, v. 153, p. 53–62, 2015.

JOSHI, Renuka. **Accuracy, Precision, Recall & F1 Score: Interpretation of Performance Measures**. Set. 2016. Disponível em: https://blog.exsilio.com/all/accuracy-precision-recall-f1-score-interpretation-of-performance-measures/. Acesso em: 15 dez. 2020.

KARPATHY, Andrej. **Convolutional Neural Networks (CNNs / ConvNets)**. Notas do curso CS231n: Convolutional Neural Networks for Visual Recognition. Disponível em: https://cs231n.github.io/convolutional-networks/. Acesso em: 20 jan. 2020.

KARPATHY, Andrej. **Image Classification**. Notas do curso CS231n: Convolutional Neural Networks for Visual Recognition. Disponível em: https://cs231n.github.io/classification/. Acesso em: 20 jan. 2020.

KARPATHY, Andrej. **Learning**. Notas do curso CS231n: Convolutional Neural Networks for Visual Recognition. Disponível em: https://cs231n.github.io/neural-networks-3/. Acesso em: 10 mar. 2020.

KARPATHY, Andrej. **Linear Classification**. Notas do curso CS231n: Convolutional Neural Networks for Visual Recognition. Disponível em: https://cs231n.github.io/linear-classify/. Acesso em: 20 jan. 2020.

KARPATHY, Andrej. **Transfer Learning**. Notas do curso CS231n: Convolutional Neural Networks for Visual Recognition. Disponível em: https://cs231n.github.io/transfer-learning/. Acesso em: 10 mar. 2020.

KINGMA, Diederik P; BA, Jimmy. Adam: A method for stochastic optimization. **arXiv preprint arXiv:1412.6980**, 2014.

KRIZHEVSKY, Alex; NAIR, Vinod; HINTON, Geoffrey. **The CIFAR-10 dataset**. 2009. Disponível em: https://www.cs.toronto.edu/~kriz/cifar.html. Acesso em: 16 fev. 2020.

KRIZHEVSKY, Alex; SUTSKEVER, Ilya; HINTON, Geoffrey E. Imagenet classification with deep convolutional neural networks. *In*: ADVANCES in neural information processing systems. [*S.l.*: *s.n.*], 2012. P. 1097–1105.

LAPUSCHKIN, Sebastian. **Model Zoo**. Abr. 2019. Disponível em: https://github.com/BVLC/caffe/wiki/Model-Zoo. Acesso em: 10 mar. 2020.

LECUN, Yann; BENGIO, Yoshua; HINTON, Geoffrey. Deep learning. **nature**, Nature Publishing Group, v. 521, n. 7553, p. 436–444, 2015. doi.org/10.1038/nature14539.

LECUN, Yann; BOTTOU, Léon; BENGIO, Yoshua; HAFFNER, Patrick. Gradient-based learning applied to document recognition. **Proceedings of the IEEE**, Ieee, v. 86, n. 11, p. 2278–2324, 1998.

LIU, Wei; ANGUELOV, Dragomir; ERHAN, Dumitru; SZEGEDY, Christian; REED, Scott; FU, Cheng-Yang; BERG, Alexander C. Ssd: Single shot multibox detector. *In*: SPRINGER. EUROPEAN conference on computer vision. [*S.l.*: *s.n.*], 2016. P. 21–37.

LUIS CABERA, José Antonio de. **El Color: Tono, saturación, brillo e iluminación**. Dez. 2011. Disponível em: `http://tonosatubrilloilu.blogspot.com/`. Acesso em: 12 mai. 2020.

LUNADEI, Loredana; RUIZ-GARCIA, Luis; BODRIA, Luigi; GUIDETTI, Riccardo. Automatic identification of defects on eggshell through a multispectral vision system. **Food and Bioprocess Technology**, Springer, v. 5, n. 8, p. 3042–3050, 2012.

MACHADO, Douglas S; PÁDUA, Flávio LC; FERNANDES, José LA. Sistema de inspeção visual automática aplicado ao controle de qualidade de ovos em linhas de produção. **CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS, Belo Horizonte**, 2009.

MERTENS, K; KEMPS, B; PERIANU, C; DE BAERDEMAEKER, J; DECUYPERE, E; DE KETELAERE, B; BAIN, M. Advances in egg defect detection, quality assessment and automated sorting and grading. *In*: IMPROVING the safety and quality of eggs and egg products. [*S.l.*]: Elsevier, 2011. P. 209–241.

MERTENS, Kristof; DE KETELAERE, Bart; KAMERS, Bram; BAMELIS, FR; KEMPS, BJ; VERHOELST, EM; DE BAERDEMAEKER, JG; DECUYPERE, EM. Dirt detection on brown eggs by means of color computer vision. **Poultry science**, Oxford University Press Oxford, UK, v. 84, n. 10, p. 1653–1659, 2005.

MOR-MUR, Montserrat; YUSTE, Josep. Emerging bacterial pathogens in meat and poultry: an overview. **Food and Bioprocess Technology**, Springer, v. 3, n. 1, p. 24, 2010.

NAKANO, Kazuhiro; USUI, Yoshihiko; MOTONAGA, Yoshitaka; MIZUTANI, Jun. Development of non-destructive detector for abnormal eggs. **IFAC Proceedings Volumes**, Elsevier, v. 34, n. 28, p. 71–76, 2001.

NARIN, Burin; BUNTAN, Sunanta; CHUMUANG, Narumol; KETCHAM, Mahasak. Crack on Eggshell Detection System Based on Image Processing Technique. *In*: IEEE. 2018 18th International Symposium on Communications and Information Technologies (ISCIT). [*S.l.*: *s.n.*], 2018. P. 1–6. doi.org/10.1109/ISCIT.2018.8587980.

NATHANCY. **Choosing the correct upper and lower HSV boundaries for color detection with 'cv::inRange' (OpenCV)**. Resposta 8 do fórum *Stackoverflow*. Jan. 2020. Disponível em: `https://stackoverflow.com/a/59906154/13105603`. Acesso em: 20 mar. 2020.

OMID, Mahmoud; SOLTANI, Mahmoud; DEHROUYEH, Mohammad Hadi; MOHTASEBI, Seyed Saeid; AHMADI, Hojat. An expert egg grading system based on machine vision and artificial intelligence techniques. **Journal of food engineering**, Elsevier, v. 118, n. 1, p. 70–77, 2013.

PESCATORE, Tony; JACOB, Jacquie. Kentucky 4-h poultry grading eggs. **University of Kentucky College of Agriculture, Food and Environment, Lexigton, KY, 40546**, mai. 2013.

PLASSON. **CLASSIFICADORA DE OVOS 90 CAIXAS AUTOMATIZADA**. Disponível em: `http://www.plasson.com.br/livestock/site/products/pattern/product/126`. Acesso em: 9 jun. 2020.

POWERS, David MW. Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. **arXiv preprint arXiv:2010.16061**, 2020.

PRIYADUMKOL, Jetsadaporn; KITTICHAIKARN, Chawalit; THAINIMIT, Somying. Crack detection on unwashed eggs using image processing. **Journal of Food Engineering**, Elsevier, v. 209, p. 76–82, 2017.

QIAN, Ning. On the momentum term in gradient descent learning algorithms. **Neural networks**, Elsevier, v. 12, n. 1, p. 145–151, 1999.

R., Fisher; S., Perkins; A., Walker; E., Wolfart. **Laplacian/Laplacian of Gaussian**. 2003. Disponível em: https://homepages.inf.ed.ac.uk/rbf/HIPR2/log.htm. Acesso em: 15 jan. 2021.

RATEKE, Thiago. **Visão Geral de Análise de Sinais Digitais, Filosofia Geral de Análise de Sinais Digitais e integração desta com Reconhecimento de Padrões**. Disponível em: http://www.lapix.ufsc.br/ensino/reconhecimento-de-padroes/gerando-padroes-analise-de-sinais-e-imagens/. Acesso em: 10 dez. 2019.

REDMON, Joseph; FARHADI, Ali. YOLO9000: better, faster, stronger. *In*: PROCEEDINGS of the IEEE conference on computer vision and pattern recognition. [*S.l.*: *s.n.*], 2017. P. 7263–7271.

RIBEIRO, Angela; GARCÍA-ALEGRE, Maria C; GUINEA, Domingo; CRISTOBAL, Gabriel. Automatic rules generation by GA for eggshell defect classification. **networks**, v. 4, p. 5, 2000.

RONNEBERGER, Olaf; FISCHER, Philipp; BROX, Thomas. U-net: Convolutional networks for biomedical image segmentation. *In*: SPRINGER. INTERNATIONAL Conference on Medical image computing and computer-assisted intervention. [*S.l.*: *s.n.*], 2015. P. 234–241.

ROSEBROCK, Adrian. **Finding extreme points in contours with OpenCV**. Linha 24 à linha 27. Abr. 2016a. Disponível em: https://www.pyimagesearch.com/2016/04/11/finding-extreme-points-in-contours-with-opencv/. Acesso em: 20 jan. 2020.

ROSEBROCK, Adrian. **Measuring size of objects in an image with OpenCV**. Mar. 2016b. Disponível em: https://www.pyimagesearch.com/2016/03/28/measuring-size-of-objects-in-an-image-with-opencv/. Acesso em: 20 mar. 2020.

ROY, Sourav Dey; DAS, Dipak Hrishi; BHOWMIK, Mrinal Kanti. Conventional and deep feature oriented quality inspection of internal defected eggs using infrared imaging. *In*: PROCEEDINGS of the 6th International Conference on Networking, Systems and Security. [*S.l.*: *s.n.*], 2019. P. 12–20. doi.org/10.1145/3362966.3362969.

RUDER, Sebastian. An overview of gradient descent optimization algorithms. **arXiv preprint arXiv:1609.04747**, 2016.

RUSSAKOVSKY, Olga *et al.* ImageNet Large Scale Visual Recognition Challenge. **International Journal of Computer Vision (IJCV)**, v. 115, n. 3, p. 211–252, 2015. DOI: 10.1007/s11263-015-0816-y.

SANTOS FERREIRA, Alessandro dos; FREITAS, Daniel Matte; SILVA, Gercina Gonçalves da; PISTORI, Hemerson; FOLHES, Marcelo Theophilo. Weed detection in soybean crops using ConvNets. **Computers and Electronics in Agriculture**, Elsevier, v. 143, p. 314–324, 2017. doi.org/10.1016/j.compag.2017.10.027.

SCIKIT-IMAGE. **Image processing in Python**. Disponível em: https://scikit-image.org/. Acesso em: 28 jul. 2020.

SEBASTIÁN, Vargas Cruz Ramiro; CECILIA, Ruiz Salvador Lourdes; CRISTINA, Navas Lema María. Merging Manual and Automated Egg Candling: A Safety and Social Solution. **Enfoque UTE**, Universidad Tecnológica Equinoccial, v. 9, n. 2, p. 70–76, 2018. doi.org/10.29019/enfoqueute.v9n2.292.

SHIMIZU, Ryota; YANAGAWA, Shusuke; SHIMIZU, Toru; HAMADA, Mototsugu; KURODA, Tadahiro. Convolutional neural network for industrial egg classification. *In*: IEEE. 2017 International SoC Design Conference (ISOCC). [*S.l.*: *s.n.*], 2017. P. 67–68. 10.1109/ISOCC.2017.8368830.

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

SLADOJEVIC, Srdjan; ARSENOVIC, Marko; ANDERLA, Andras; CULIBRK, Dubravko; STEFANOVIC, Darko. Deep neural networks based recognition of plant diseases by leaf image classification. **Computational intelligence and neuroscience**, Hindawi, v. 2016, 2016. doi.org/10.1155/2016/3289801.

SOBRADO MALPARTIDA, Eddie Ángel. Sistema de visión artificial para el reconocimiento y manipulación de objetos utilizando un brazo robot. Pontificia Universidad Católica del Perú, 2003.

STANFORD. **Lecture 11 Detection and Segmentation**. 1 video (74 min). STANFORD UNIVERSITY SCHOOL OF ENGINEERING. Ago. 2017a. Disponível em: https://www.youtube.com/watch?v=nDPWywWRIRo&t=2041s&ab_channel=StanfordUniversitySchoolofEngineering. Acesso em: 15 fev. 2020.

STANFORD. **Lecture 2 Image Classification**. 1 video (90 min). STANFORD UNIVERSITY SCHOOL OF ENGINEERING. Ago. 2017b. Disponível em: https://www.youtube.com/watch?v=0oUX-nOEjG0%5C&list=PL3FW7Lu3i5JvHM8ljYj-zLfQRF3EO8sYv%5C&index=3%5C&t=0s. Acesso em: 15 fev. 2020.

STANFORD. **Lecture 3 Loss Functions and Optimization**. 1 video (74 min). STANFORD UNIVERSITY SCHOOL OF ENGINEERING. Ago. 2017c. Disponível em: https://www.youtube.com/watch?v=h7iBpEHGVNc%5C&list=PL3FW7Lu3i5JvHM8ljYj-zLfQRF3EO8sYv%5C&index=3. Acesso em: 16 fev. 2020.

STANFORD. **Lecture 9 CNN Architectures**. 1 video (77 min). STANFORD UNIVERSITY SCHOOL OF ENGINEERING. Ago. 2017d. Disponível em: https://www.youtube.com/watch?v=DAOcjicFr1Y%5C&list=PL3FW7Lu3i5JvHM8ljYj-zLfQRF3EO8sYv%5C&index=9. Acesso em: 20 mar. 2020.

STIVANELLO, Maurício Edgar; ROLOFF, Mário Lucio; STEMMER, Marcelo Ricardo. **TEM –Sistemas de Visão, parte 01: introdução aos sistemas de visão e processamento de imagens digitais**. [*S.l.*: *s.n.*], nov. 2019.

SZEGEDY, Christian; LIU, Wei; JIA, Yangqing; SERMANET, Pierre; REED, Scott; ANGUELOV, Dragomir; ERHAN, Dumitru; VANHOUCKE, Vincent; RABINOVICH, Andrew. Going deeper with convolutions. *In*: PROCEEDINGS of the IEEE conference on computer vision and pattern recognition. [*S.l.*: *s.n.*], 2015. P. 1–9.

TIU, Ekin. **Metrics to Evaluate your Semantic Segmentation Model**. Ago. 2019. Disponível em: https://towardsdatascience.com/metrics-to-evaluate-your-semantic-segmentation-model-6bcb99639aa2. Acesso em: 15 dez. 2020.

TORRALBA, Antonio; RUSSELL, Bryan C; YUEN, Jenny. Labelme: Online image annotation and applications. **Proceedings of the IEEE**, IEEE, v. 98, n. 8, p. 1467–1484, 2010. doi.org/10.1109/JPROC.2010.2050290.

TRIANTAPHYLLOU, Evangelos. Multi-criteria decision making methods. *In*: MULTI-CRITERIA decision making methods: A comparative study. [*S.l.*]: Springer, 2000. P. 5–21.

VASILEVA, AV; GORBUNOVA, EV; VASILEV, AS; PERETYAGIN, VS; CHERTOV, AN; KOROTAEV, VV. Assessing exterior egg quality indicators using machine vision. **British poultry science**, Taylor & Francis, v. 59, n. 6, p. 636–645, 2018.

VON WANGENHEIM, Aldo. **Aula 02 Domínio do Valor em Visão Computacional (v.2)**. 1 vídeo (82 min). Set. 2019a. Disponível em: https://www.youtube.com/watch?v=4UIdPigzw_8%5C&list=PLmDIGfkfgKy1SBjXA0kBk4DAhIaN1vQOS%5C&index=2%20,%20http://www.lapix.ufsc.br/ensino/reconhecimento-de-padroes/gerando-padroes-analise-de-sinais-e-imagens/. Acesso em: 20 set. 2019.

VON WANGENHEIM, Aldo. **Aula 04 - Convolução, Morfologia Matemática e Filtros**. 1 video (99 min). Set. 2019b. Disponível em: https://www.youtube.com/watch?v=mSb_rGWfeWM%5C&list=PLmDIGfkfgKy1SBjXA0kBk4DAhIaN1vQOS%5C&index=4. Acesso em: 20 out. 2019.

VON WANGENHEIM, Aldo. **Aula 05 Detecção de Bordas**. 1 video (71 min). Set. 2019c. Disponível em: https://www.youtube.com/watch?v=4v8Pyn45Els%5C&list=PLmDIGfkfgKy1SBjXA0kBk4DAhIaN1vQOS%5C&index=5. Acesso em: 20 set. 2019.

VON WANGENHEIM, Aldo. **Deep Learning::Introdução ao Novo Coneccionismo**. Disponível em: http://www.lapix.ufsc.br/ensino/visao/visao-computacionaldeep-learning/deep-learningintroducao-ao-novo-coneccionismo/. Acesso em: 10 jan. 2020.

VON WANGENHEIM, Aldo. **ResNet**. Disponível em: http://www.lapix.ufsc.br/ensino/visao/visao-computacionaldeep-learning/deep-learningreconhecimento-de-imagens#ResNet. Acesso em: 3 mar. 2020.

VON WANGENHEIM, Aldo. **Visão Computacional**. Disponível em: http://www.lapix.ufsc.br/ensino/visao/. Acesso em: 18 set. 2019.

WARANUSAST, Rattapoom; INTAYOD, Pongsakorn; MAKHOD, Donlaya. Egg size classification on Android mobile devices using image processing and machine learning. *In*: IEEE. 2016 Fifth ICT International Student Project Conference (ICT-ISPC). [*S.l.*: *s.n.*], 2016. P. 170–173.

WEE, ROLANDO Y. **Top Egg Producing Countries In The World**. 2019. Disponível em: worldatlas.com/articles/top-egg-producing-countries-in-the-world.html.

WIKIPEDIA. **Modelo de color HSV**. Disponível em: https://es.wikipedia.org/wiki/Modelo_de_color_HSV. Acesso em: 28 jul. 2020.

YAMASHITA, Rikiya; NISHIO, Mizuho; DO, Richard Kinh Gian; TOGASHI, Kaori. Convolutional neural networks: an overview and application in radiology. **Insights into imaging**, Springer, v. 9, n. 4, p. 611–629, 2018.

YOSINSKI, Jason; CLUNE, Jeff; BENGIO, Yoshua; LIPSON, Hod. How transferable are features in deep neural networks? **CoRR**, abs/1411.1792, 2014. arXiv: 1411.1792. Disponível em: http://arxiv.org/abs/1411.1792. Acesso em: 10 mar. 2020.

ZALHAN, Mohd Zin; SYARMILA, Sameon Sera; NAZRI, Ismail Mohd; TAHA, Ismail Mohd. Vision-based egg grade classifier. *In*: IEEE. 2016 International Conference on Information and Communication Technology (ICICTM). [*S.l.*: *s.n.*], 2016. P. 31–35.

ZEILER, Matthew D; FERGUS, Rob. Visualizing and understanding convolutional networks. *In*: SPRINGER. EUROPEAN conference on computer vision. [*S.l.*: *s.n.*], 2014. P. 818–833.

ZEILER, MD; FERGUS, R. Visualizing and understanding convolutional networks. arXiv 2013. **arXiv preprint arXiv:1311.2901**, 2019.

ZHANG, MR; YE, JS; LI, XY; ZHANG, ZL; HAN, JJ; WANG, YH. A preliminary inquiry into causes of dark spots in egg shell. **Food & Machinery**, p. 05, 2016.

ZHONG, Zhun; ZHENG, Liang; KANG, Guoliang; LI, Shaozi; YANG, Yi. Random Erasing Data Augmentation. **CoRR**, abs/1708.04896, 2017. arXiv: 1708.04896. Disponível em: http://arxiv.org/abs/1708.04896. Acesso em: 10 mar. 2020.

— —

Apêndices

APÊNDICE A – USANDO *PYTORCH* PARA *FAST.AI*

A biblioteca *Fastai* (HOWARD; GUGGER, 2020) fornece muitas funções úteis que permitem construir redes neurais de maneira rápida e fácil para treinar modelos. *Fastai* está no *PyTorch*. O *PyTorch* está crescendo rapidamente em uma biblioteca extremamente popular e funciona muito mais rápido em comparação com o *TensorFlow*. Para este projeto, a biblioteca *Fastai* foi usada para treinar a rede, usando o *Tranfer learning*, alguns dos conceitos básicos a serem considerados neste processo são descritos abaixo:

Um aplicativo em *fastai* usa 4 etapas básicas:

a. Criar um *DataLoaders*.

b. Criar um *Learner*.

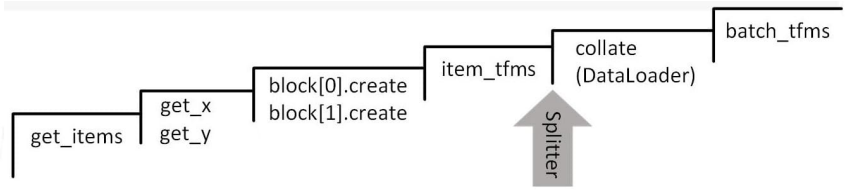
c. Empregar um método de ajuste.

d. Fazer previsões e analisar os resultados.

Criar um *DataLoaders*

Numa primeira fase é necessário definir o tipo de dados a utilizar e como eles são estruturados. Uma maneira de fazer isso é usando uma *API* de alto nível fornecida por *fastai*, conhecida como *Datablock* (FASTAI, 2020). A estrutura geral desta *API* é apresentada na Figura 107. O que a *API* de *Datablock* faz é criar um conjunto de dados para treinamento e validação, este método é responsável por obter as imagens, associar as classes, dividir o conjunto de treinamento e validação e aplicar transformações. Feito este processo, é concatenado tudo em um objeto chamado *DataLoader*, que é responsável por pegar conjuntos de imagens e organizá-las em lotes (*batch*).

Figura 107 – Propriedades da *API Datablock*.



Fonte – Adaptado de (HOWARD, 2020b).

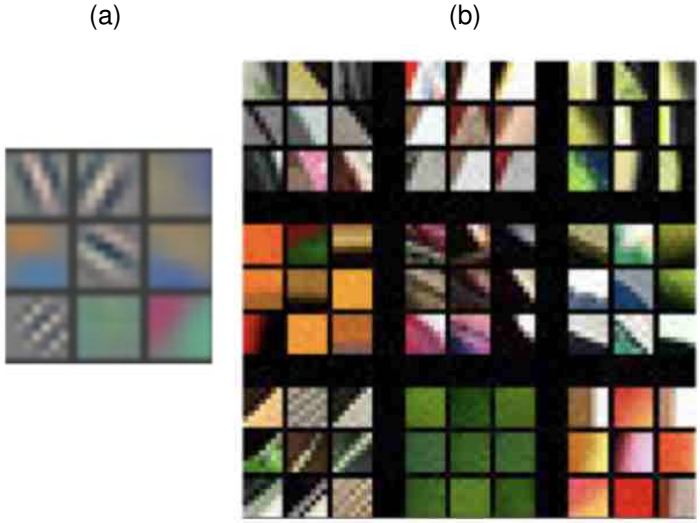
Criar um *Learner*

O *vision.learner* é o módulo que define o método *cnn_learner*, para obter facilmente um modelo adequado para *tranfer learning (learner)*. Este modelo preserva todas as camadas convolucionais com seus pesos previamente treinados no *ImageNet*, remove a última camada e cria duas novas matrizes com números aleatórios e *Relu* no meio. A próxima etapa envolve o treinamento dessas duas matrizes com o

conjunto de dados de destino, sem modificar os outros pesos associados às camadas iniciais, porque eles já foram treinados. Basicamente, as primeiras camadas destinam-se a reconhecer características gerais dos objetos. Este processo é descrito em mais detalhes abaixo:

a. *Primeira camada*: uma visualização simplificada da primeira camada de uma *CNN* treinada no *ImageNet* é mostrada na Figura 108.

Figura 108 – Visualização da primeira camada de uma *CNN* treinada com *ImageNet*.

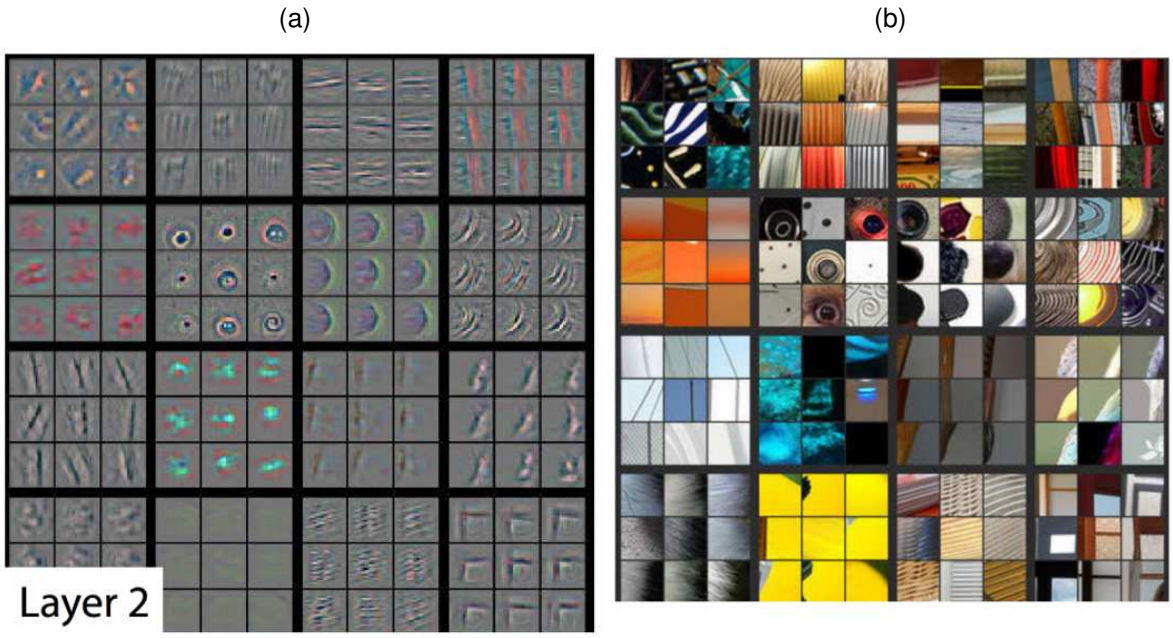


Fonte – (ZEILER, M.; FERGUS, 2019).

A Figura 108a apresenta nove exemplos dos coeficientes reais da primeira camada. Eles operam em grupos de pixels próximos um do outro. Portanto, este primeiro, por exemplo encontra grupos de pixels que têm uma pequena linha diagonal, o segundo encontra uma linha diagonal na outra direção, o terceiro encontra gradientes que vão do amarelo ao azul e assim por diante. Eles são pequenos filtros muito simples (HOWARD, 2019).

b. *Segunda camada*: a camada 2 mostrada na Figura 109, pega os resultados dos filtros da primeira camada e executa uma segunda camada de cálculo. Como pode ser visto na Figura 109a, os filtros aprendem a criar diferenças específicas, como, por exemplo, o filtro no canto inferior direito, aprendeu a criar algo que diferencia os cantos superiores esquerdos, outros aprendem a encontrar curvas, pequenos círculos, etc. Na camada um, tem-se filtros que podem encontrar apenas uma linha e, na camada 2, pode-se encontrar itens com duas linhas unidas ou uma linha repetida. Na Figura 109b, são mostrados exemplos de bits reais das imagens que ativaram esses filtros, onde por exemplo, o filtro no canto inferior direito foi bom para encontrar os cantos das janelas (HOWARD, 2019).

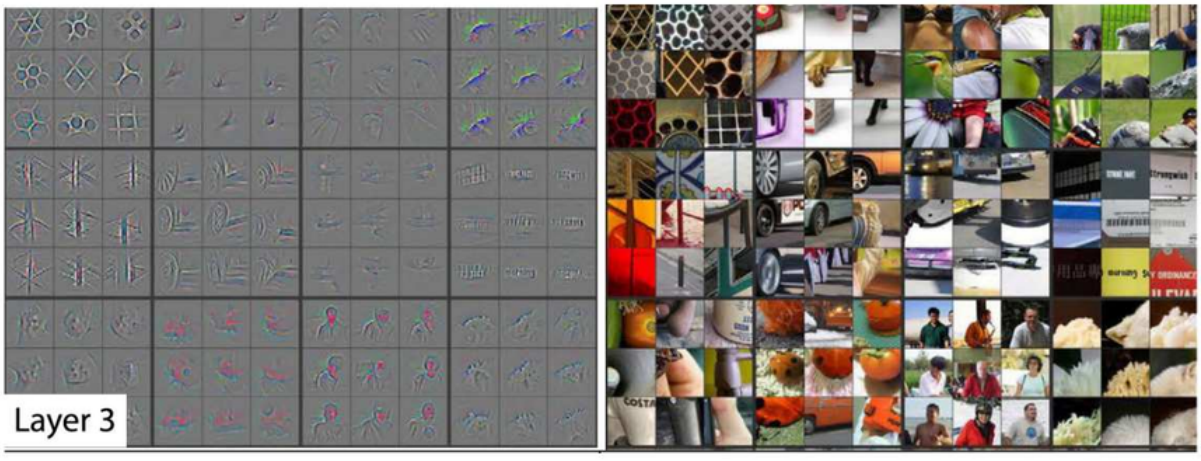
Figura 109 – Visualização da segunda camada de uma *CNN* treinada com *ImageNet*.



Fonte – (ZEILER, M.; FERGUS, 2019).

c. *Terceira camada:* nesta camada, já é possível encontrar padrões repetitivos de objetos bidimensionais ou pode-se encontrar coisas que se juntam. A Camada 3 consegue pegar todas as coisas da Camada 2 e combiná-las, como mostra a Figura 110.

Figura 110 – Visualização da terceira camada de uma *CNN* treinada com *ImageNet*.

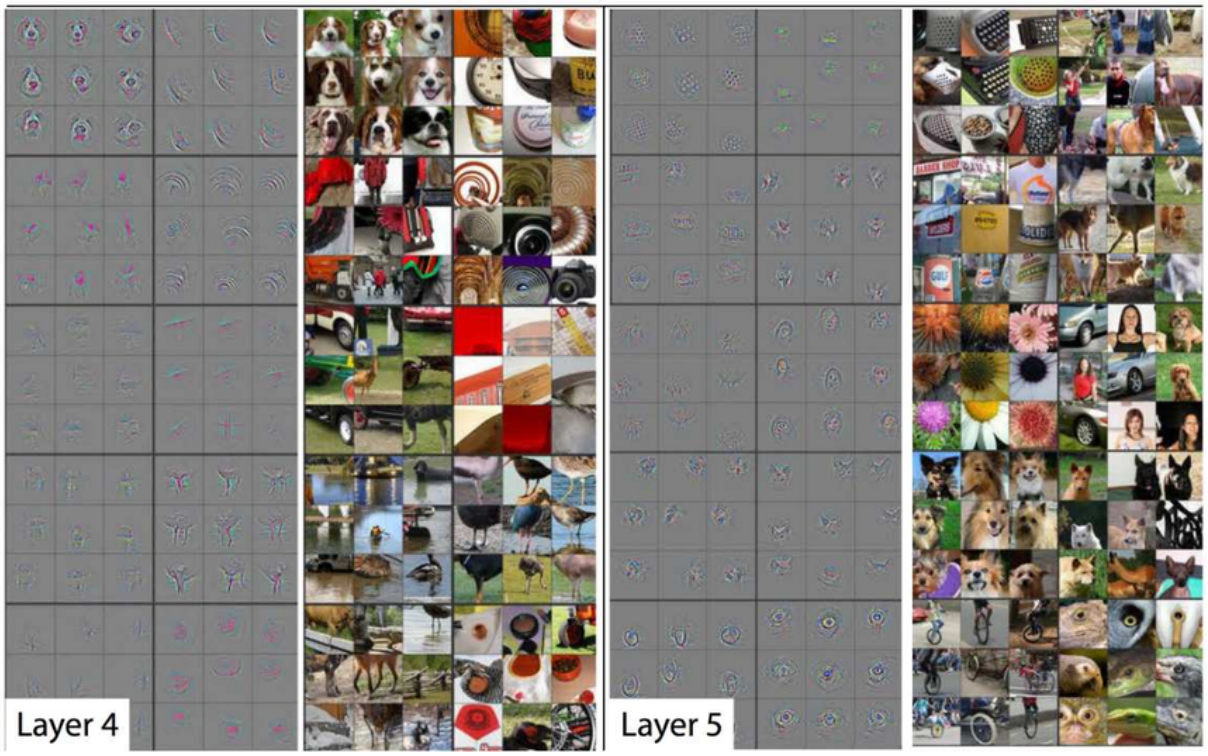


Fonte – (ZEILER, M.; FERGUS, 2019).

d. *camada quatro e cinco:* a camada 4 pode pegar todas as coisas da camada 3 e combiná-las. Como por exemplo, pode-se obter algo que pode encontrar rostos de cães ou pés de pássaros. Na camada 5, tem-se algo que pode encontrar

globos oculares de pássaros e lagartos ou rostos de raças específicas de cães, etc.

Figura 111 – Visualização da camada 4 e 5 de uma *CNN* treinada com *ImageNet*.



Fonte – (ZEILER, M.; FERGUS, 2019).

Cada vez que o número de camadas é aumentado, o modelo pode encontrar características cada vez mais particulares dos objetos. A partir desses modelos, alguns padrões repetitivos encontrados nas primeiras camadas podem ser reutilizados, e é por isso que essas camadas são *"congeladas" (Freezing)* porque não precisam ser modificadas e apenas as duas últimas matrizes são trabalhadas inicialmente.

Empregar um método de ajuste

Para adaptar o modelo, um método chamado *fit_one_cycle* é usado. Isso define basicamente quantas vezes irá percorrer todo o conjunto de dados, ou seja, quantas vezes o modelo verá os dados para aprender com ele. Nesse estágio, é necessário cuidar do *overfitting*, pois, se o modelo vê as mesmas imagens muitas vezes, pode diminuir seu poder de generalização.

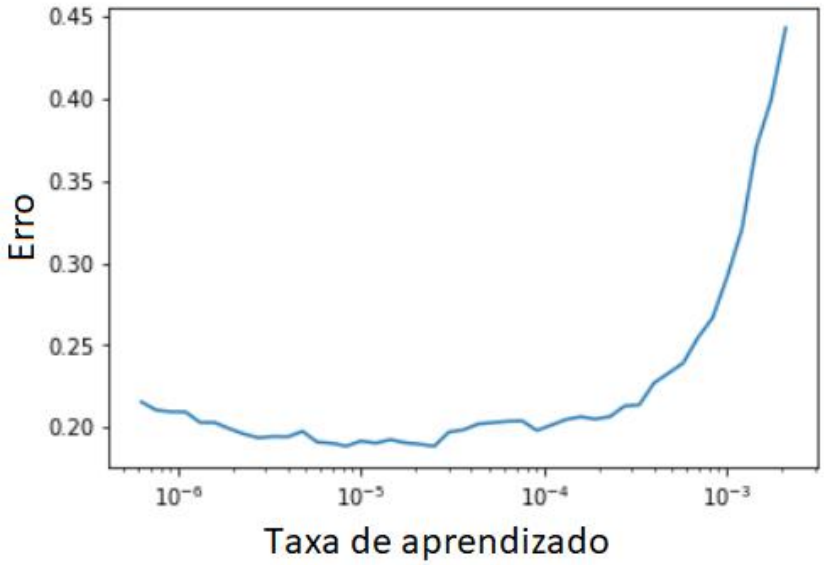
Quando o modelo é adaptado ao novo conjunto de dados de destino, funcionará bem porque a rede é capaz de reconhecer características gerais que são comumente encontradas em imagens como linhas, círculos, entre outras. Outra característica desse modelo é que seu treinamento será relativamente rápido, porque o que foi feito foi adicionar algumas camadas adicionais no final e apenas essas camadas foram treinadas. No entanto, esse modelo pode ser aprimorado e isso é alcançado durante o processo

de *Fine-Tuning*.

Na etapa de *Fine-Tuning*, toda a rede é *descongelada* para unir todo o processo. Nesta etapa, as camadas adicionadas no final provavelmente precisam de mais treinamento, e as camadas no início, por exemplo, bordas diagonais provavelmente não precisam de muito treinamento. É por isso que o modelo é dividido em algumas seções, que terão taxas de aprendizado diferentes. Isso é conhecido como treinamento de otimização de hiperparâmetros, que é facilmente fornecido por modelos como o *ResNet*.

Para definir quais taxas de aprendizado são mais adequadas para o modelo, é usado um método de ajuste conhecido como *"lr_find"*. Este método irá desenhar um gráfico como o mostrado na Figura 112, no qual o eixo *x* determina o que acontece à medida que a taxa de aprendizado aumenta e o eixo *y* mostra qual é a perda. Como pode ser visto, uma vez que a taxa de aprendizado excede 1e-4, a perda piora. Uma taxa de aprendizado deve ser escolhida muito antes do modelo piorar, por exemplo, 1e-6. Como não faz sentido treinar todas as camadas a esse ritmo, porque é sabido que as camadas posteriores funcionavam bem antes quando eram treinadas muito mais rápido. Então, o que realmente é feito é passar um intervalo de taxas de aprendizado usando *"max_lr = slice (1e-6,1e-4)"*, que treina as primeiras camadas a uma velocidade de aprendizado de 1e-6 e, as últimas camadas na velocidade de 1e-4. Todas as outras camadas estão distribuídas igualmente entre estes dois valores (HOWARD, 2019).

Figura 112 – Método de ajuste "*lr_find*".



Fonte – (HOWARD, 2019).

É necessário esclarecer que uma taxa de aprendizado diferente não é atribuída para cada camada, uma taxa de aprendizado diferente é atribuída a cada "grupo de camada". As duas camadas adicionais criadas no início do modelo são conhecidas

como um grupo e o restante do modelo é dividido ao meio em dois grupos de camadas. Por padrão, três grupos de camadas serão obtidos. Se a função de *slide(1e-5, 1e-3)* for executada, será obtida uma taxa de aprendizado de 1e-5 para o primeiro grupo de camadas, 1e-4 para o segundo e 1e-3 para o terceiro.

A última versão do *Fastai* lançada o 21 de agosto de 2020 permite que todo esse processo seja feito com um único método chamado (*fine_tune*). A configuração geral deste método é mostrada na Figura 113, onde na primeira etapa (1) o método *freeze* é chamado, o que faz com que apenas os pesos da última camada sejam atualizados pelo otimizador, depois é chamado o método *fit_one_cycle* descrito acima, este método por defeito é executado apenas para um *loop (freeze_epochs = 1)* com uma taxa de aprendizado inicial de *(lr = 0,002)* que é o que geralmente é usado, no entanto, isso pode ser alterado a qualquer momento. Depois disso, a mesma taxa de aprendizado não é mais necessária, então a taxa inicial de aprendizado é dividida pela metade. Finalmente, toda a rede é descongelada *(unfreeze)* para que todos os parâmetros possam ser calculados, estes parâmetros são ajustados por algumas épocas definidas no momento da chamada do método. Todo esse processo permite o treinamento da rede.

Figura 113 – Método *Fine_tune* de *Fastai*.

```
Signature: learn.fine_tune(epochs, base_lr=0.002, freeze_epochs=1, lr_mult=100, pct_start=0.3,
div=5.0, lr_max=None, div_final=100000.0, wd=None, moms=None, cbs=None, reset_opt=False)
Source:
@patch
@log_args(but_as=Learner.fit)
@delegates(Learner.fit_one_cycle)
def fine_tune(self:Learner, epochs, base_lr=2e-3, freeze_epochs=1, lr_mult=100,
              pct_start=0.3, div=5.0, **kwargs):
    "Fine tune with `freeze` for `freeze_epochs` then with `unfreeze` from `epochs` using
discriminative LR"
  1 self.freeze()
  2 self.fit_one_cycle(freeze_epochs, slice(base_lr), pct_start=0.99, **kwargs)
  3 base_lr /= 2
  4 self.unfreeze()
  5 self.fit_one_cycle(epochs, slice(base_lr/lr_mult, base_lr), pct_start=pct_start, div=div,
**kwargs)
File:      /usr/local/lib/python3.6/dist-packages/fastai/callback/schedule.py
Type:      method
```

Fonte – Criado pelo autor.

Fazer previsões e analisar os resultados

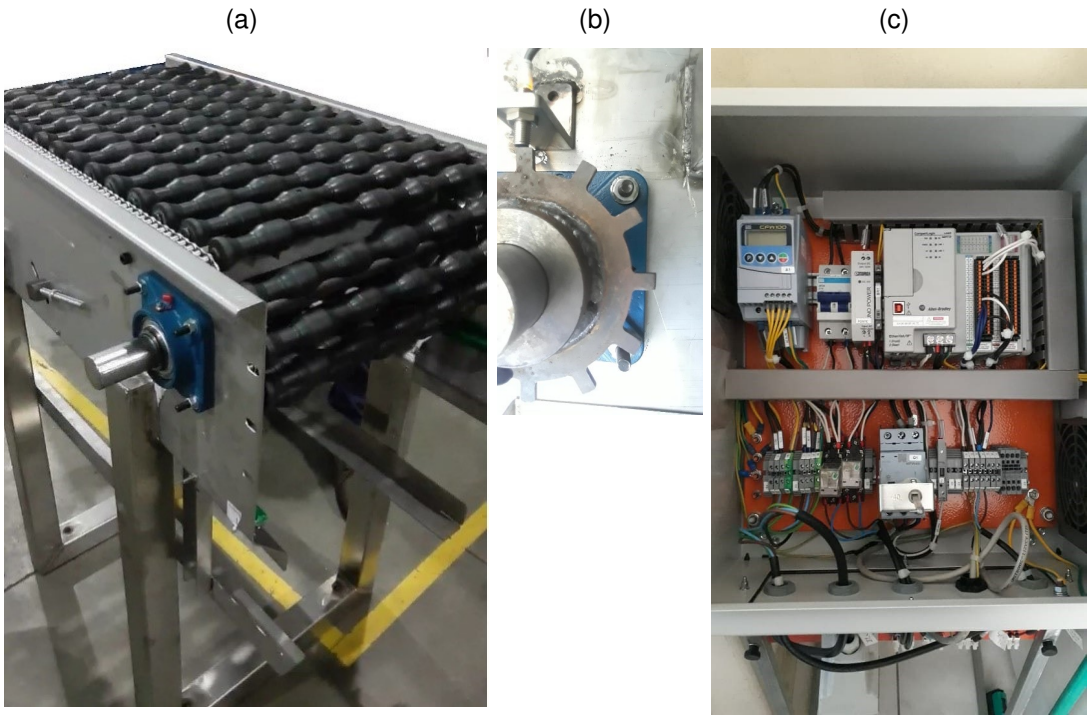
Para analisar os resultados, *fastai* fornece vários métodos simples como *learn.show_results()* que permite analisar as classes previstas pelo modelo em conjunto com as classes reais, também é possível fazer previsões tanto para um conjunto de imagens quanto para imagens individuais.

APÊNDICE B – IMPLEMENTAÇÃO DO PROTÓTIPO DE INSPEÇÃO

O protótipo de inspeção visual de ovos foi composto por dois módulos: um módulo que permite a movimentação dos ovos e um módulo associado ao compartimento fechado para a captura das imagens. O módulo de movimentação foi fornecido pela empresa *Plasson Ltda.*, composto pelos seguintes componentes:

- Conjunto de rolos cônicos acoplados em uma corrente rotativa num dispositivo metálico (Figura 114a).
- Um motor que permite a movimentação da corrente rotativa.
- Uma roda dentada associada ao eixo do motor e um sensor de proximidade que envia pulsos para cada dente (Figura 114b). A distância de um dente ao outro é equivalente a uma volta de 360° de cada rolo na corrente rotativa.
- Um painel de controle com um controlador lógico programável *CompactLogix L24ER QBFC1B* que permite a conexão e o controle de todos os dispositivos eletrônicos.

Figura 114 – (a) Rolos cônicos; (b) Roda dentada e sensor associado ao eixo do motor; (c) Painel de controle.

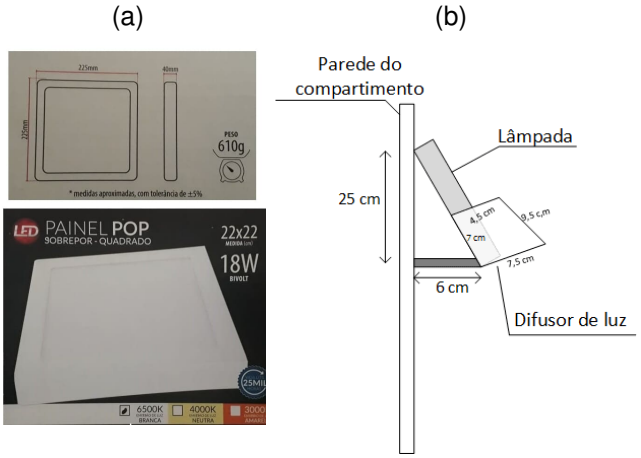


Fonte – O autor.

O segundo módulo tem como objetivo manter uma iluminação uniforme para a captura das imagens, a cor no interior do compartimento foi selecionada com base em

informações levantadas na revisão da literatura, onde foram utilizadas paredes brancas e pretas, portanto, ambas as opções foram testadas, obtendo-se melhores resultados com paredes brancas que permitem um melhor contraste nas imagens. As lâmpadas LED foram selecionadas para iluminação superior dentro do módulo, pois garantem uma iluminação difusa ideal para este caso. Para isso, foram adquiridos dois painéis LED quadrados de 18 W com temperatura de cor de 6500K (Figura 115a) que foram colocados em duas das laterais do compartimento. Para reduzir os reflexos diretos da luz nos ovos, esses painéis foram colocados diagonalmente voltados para cima conforme mostrado na Figura 115b, com um escudo na parte inferior que funcionava como um difusor de luz.

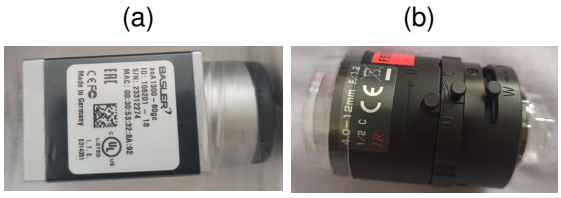
Figura 115 – (a) Referência lâmpadas; (b) Localização das lâmpadas no compartimento.



Fonte – O autor.

Para a seleção da câmera, considerou-se que deveria ser uma câmera projetada para ambientes industriais capaz de captar imagens com alta resolução em alta velocidade e formato colorido, visto que na literatura diversos trabalhos usaram o espaço de cores *HSV* para realizar a detecção de defeitos. Por este motivo, foi selecionada uma câmera industrial *Basler acA1300-60 gc* (Figura 116a) com lente varifocal de *4-12 mm e zoom manual de 1/2'* (Figura 116b) que adquire imagens no espaço de cores RGB a 60 fps com resolução de (1280 x 1024) px.

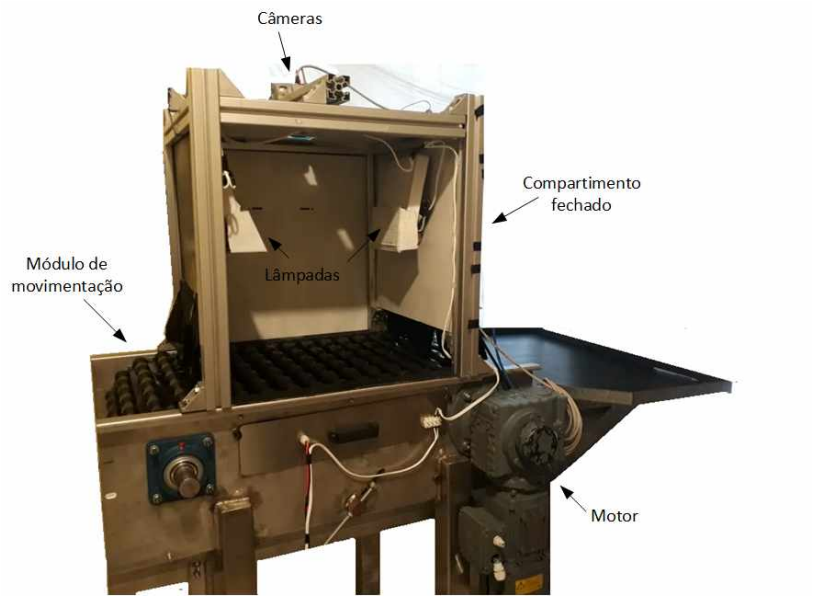
Figura 116 – (a) Câmera; (b) Lente.



Fonte – O autor.

Foram usados dois computadores, um da marca *Inspiron* 3583 com processador *Intel Core (TM) i7-8565U a 1,80 Ghz* com 8,00 GB de RAM com placa *Radeon* 520 e o segundo computador da marca *ASUS* com processador *Intel Core (TM) I5 -9300 H CPU @* 2,40 GHz com 8,00 GB de RAM com placa *NVIDIA GeForce GTX 1650*. As Figuras 117 e 118 mostram o protótipo final acoplando os dois módulos para realizar a inspeção dos ovos.

Figura 117 – Protótipo desenvolvido para inspeção de ovos.



Fonte – O autor.

Figura 118 – Protótipo desenvolvido para inspeção de ovos.

(a) (b)

Fonte – O autor.

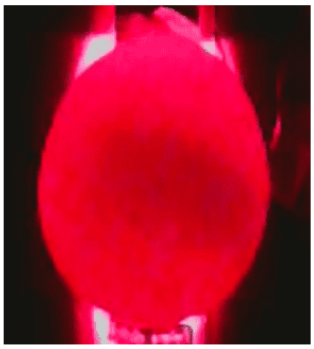
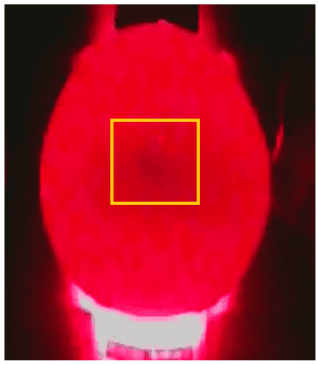
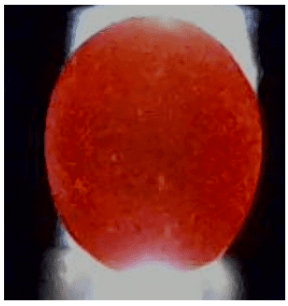
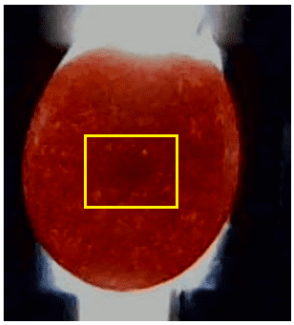
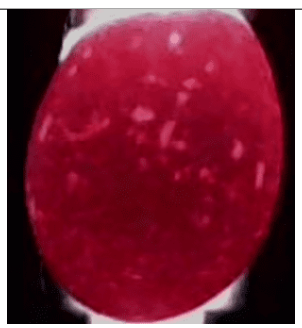
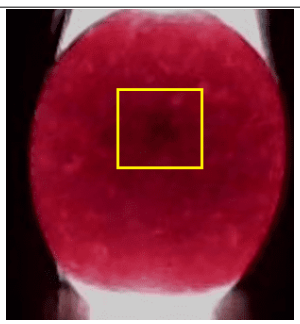
ILUMINAÇÃO INFERIOR

Um dos requisitos iniciais do projeto foi encontrar defeitos associados ao sangue interno nos ovos. Para isso o ovo tem que ser iluminado por baixo para observar

detalhes em seu interior. A escolha da iluminação adequada para este fim foi um dos principais desafios do trabalho, pois na literatura não foram encontradas referências específicas de como a iluminação era utilizada nas técnicas de ovoscopia. No entanto, alguns trabalhos mencionaram que utilizavam lâmpadas halógenas maiores ou iguais a 50 W.

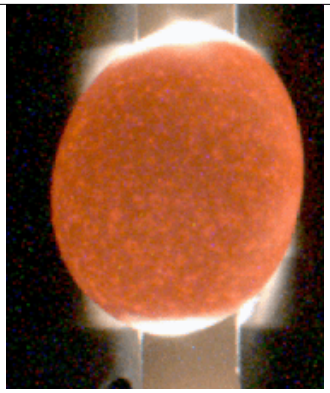
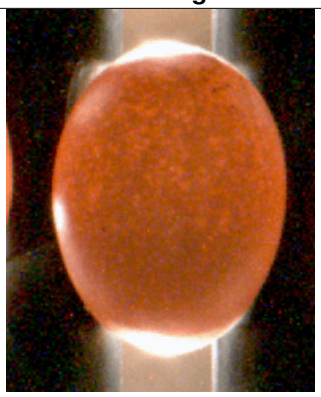
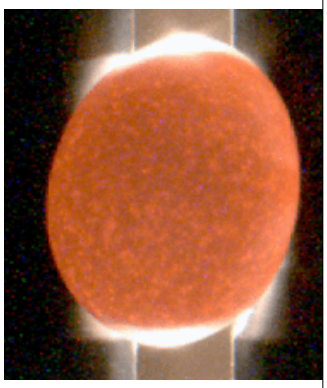
Para encontrar a iluminação certa para este projeto, foram realizadas duas fases de teste com diferentes iluminadores de diferentes potências e cores. Numa primeira fase foram testados três iluminadores com (0,5 W, 20 W e 50 W). As imagens foram capturadas com uma *Logitech C270 Webcam* em ovos vermelhos modificados manualmente, aos quais 5 mm de sangue de galinha foram adicionados com uma seringa de 23 G x 25 mm, ovos vermelhos foram escolhidos porque dentre as duas categorias para avaliar (vermelho e branco) os defeitos dentro desses ovos são mais difíceis de detectar, portanto, se bons resultados forem obtidos nesta categoria, também serão obtidos nos ovos brancos. Os resultados por cada iluminador nesta primeira fase para um ovo normal e um ovo com sangue interno são mostrados no Quadro 14.

Quadro 14 – Testes de iluminação primeira fase.

	Normal	**Com sangue**	**Observações**
Anel LED (0.5 W) Temperatura de cor 1000K. Componente RED.			Com uma temperatura de cor de 1000K, é difícil diferenciar entre a gema e o sangue dentro do ovo.
Lâmpada LED dicróica (20 W) Temperatura de cor 6000K.			Resultados promissores são obtidos com 20 W, no entanto, manchas de sangue ainda não são fáceis de diferenciar.
Lâmpada LED dicróica (50 W) Temperatura de cor 4000K.			Com 50 W é possível diferenciar a mancha de sangue, porém, devido à baixa temperatura da cor, a gema torna-se perceptível, o que pode causar erros na detecção.

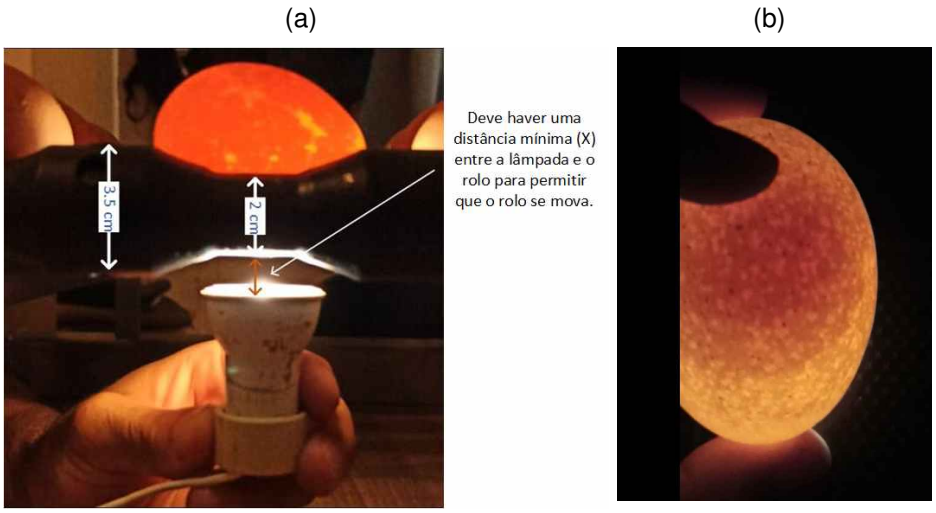
A partir dos resultados da primeira fase, constatou-se que melhores resultados são obtidos com lâmpadas que apresentassem potência maior que 20 W com temperatura de cor maior ou igual a 6000K. Com essas considerações, foi feita uma segunda fase de teste, em onde foram testadas duas lâmpadas de 25 W e 50 W. Neste caso, as capturas foram feitas com a câmera industrial *Basler acA1300-60 gc* em ovos normais e ovos reais que continham sangue dentro deles classificados por um especialista. Os resultados são apresentados no Quadro 15.

Quadro 15 – Testes de iluminação segunda fase.

	Normal	**Com sangue**	**Observações**
Lâmpada LED dicróica (3 W equivalente a 25 W) Temperatura de cor 6500K.			Neste caso, há manchas de sangue que não são tão concentradas como no caso dos testes iniciais. Nestes casos não foi possível diferenciar o defeito no interior do ovo.
Lâmpada LED dicróica (7 W equivalente a 50 W) Temperatura de cor 6000k.			Resultados semelhantes são obtidos como acima. Uma consideração importante neste caso é que a temperatura de cor desempenha um papel fundamental, pois uma lâmpada com uma temperatura de 6500K e 25W obtém resultados semelhantes a uma lâmpada de 6000K com o dobro da potência.

Como pode ser visto, não foi possível gerar uma diferenciação clara entre um ovo normal e um ovo com defeito interno, e isso se deve ao fato de o ovo não estar sendo iluminado diretamente, ou seja, há uma distância entre o foco da lâmpada e o ovo conforme mostrado na Figura 119a. Essa distância gera vazamentos de luz que não só reduzem o contraste da imagem, mas também fazem com que o interior do ovo não seja iluminado corretamente. Para testar essa teoria, foi analisado um ovo com sangue interno, colocando-o diretamente no foco da lâmpada, evitando o vazamento de luz, que é o que normalmente se faz nas técnicas de ovoscopia manual, o resultado é mostrado na Figura 119b onde é possível diferenciar o defeito dentro do ovo.

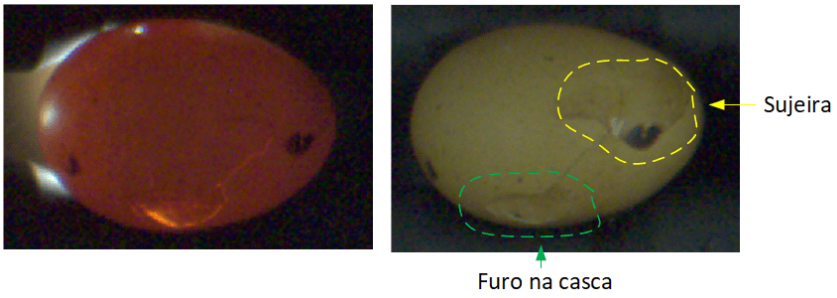
Figura 119 – Testes de iluminação.



Fonte – O autor.

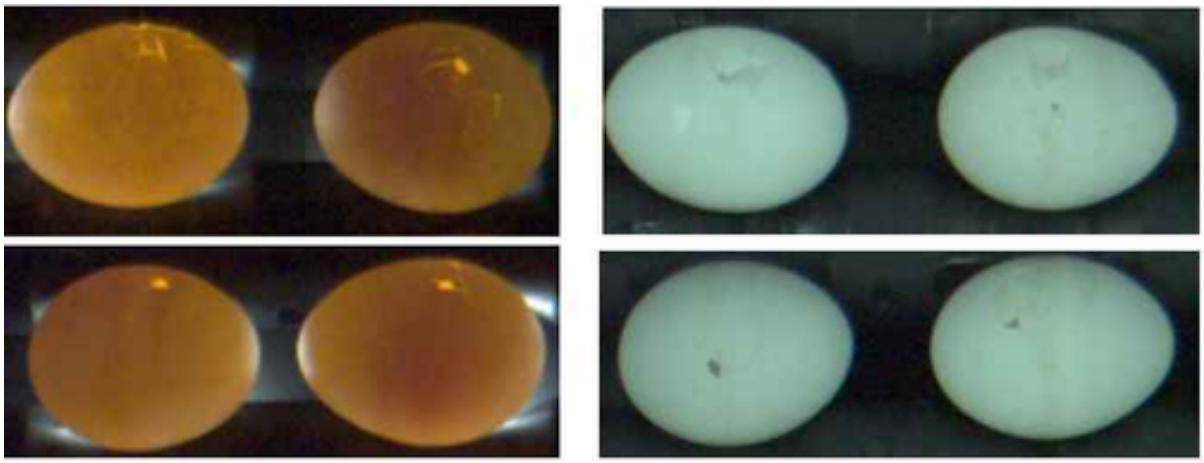
Por essa razão, defeitos de sangue interno tiveram que ser considerados para trabalhos futuros, uma vez que a infraestrutura dos rolos utilizados no projeto não permitia modificações físicas que pudessem gerar uma melhor captura dos defeitos internos nos ovos. Outra descoberta importante deste segundo teste sob iluminação inferior foi que os defeitos externos, como buracos e fissuras na casca, eram mais fáceis de diferenciar em comparação com a iluminação superior. Um exemplo disso é mostrado na Figura 120, onde fica evidente que as fissuras, embora possam ser identificadas com iluminação superior, são difíceis de diferenciar computacionalmente de outros defeitos externos como sujeira. Não é o caso da imagem com iluminação inferior em que a fissura permite a passagem da luz gerando uma notável diferenciação do defeito em relação aos demais. A diferença é ainda mais perceptível nos ovos brancos (Figura 121), onde fissuras e buracos são quase imperceptíveis apenas com iluminação superior. Por esses motivos, a iluminação inferior foi implementada com lâmpadas dicróicas de 25 W em uma base com orifícios localizados sob cada ovo, conforme mostrado na Figura 122.

Figura 120 – Análise de fissuras com iluminação inferior e superior.



Fonte – O autor.

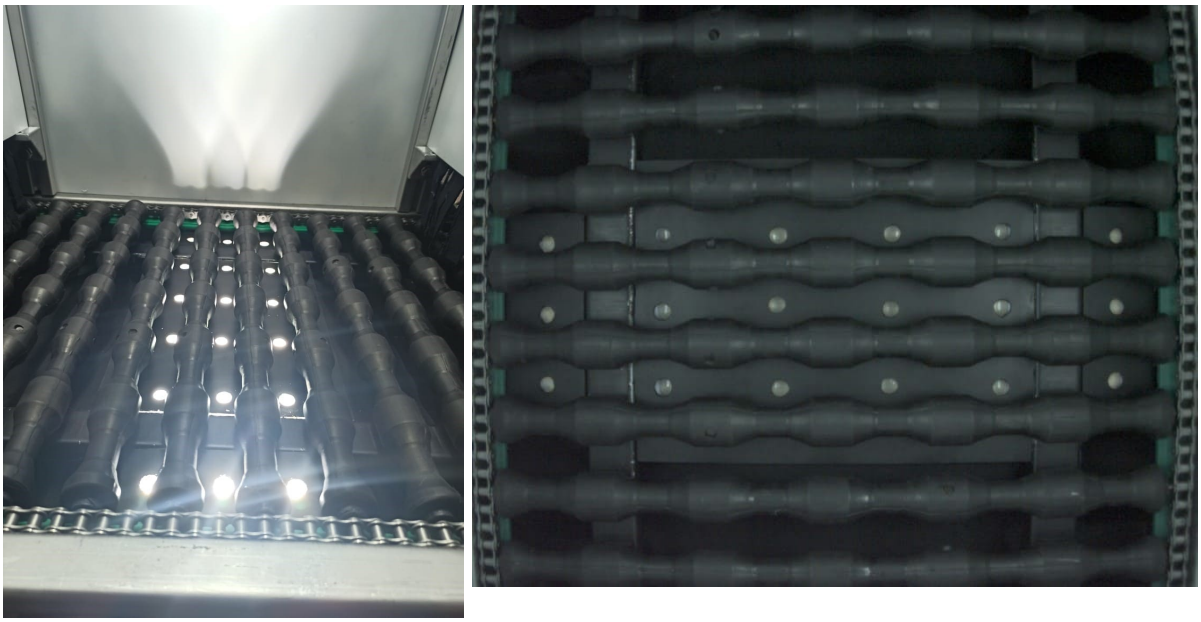
Figura 121 – Análise de fissuras com iluminação inferior e superior.



Fonte – O autor.

Figura 122 – Localização das lâmpadas inferiores.

(a) (b)

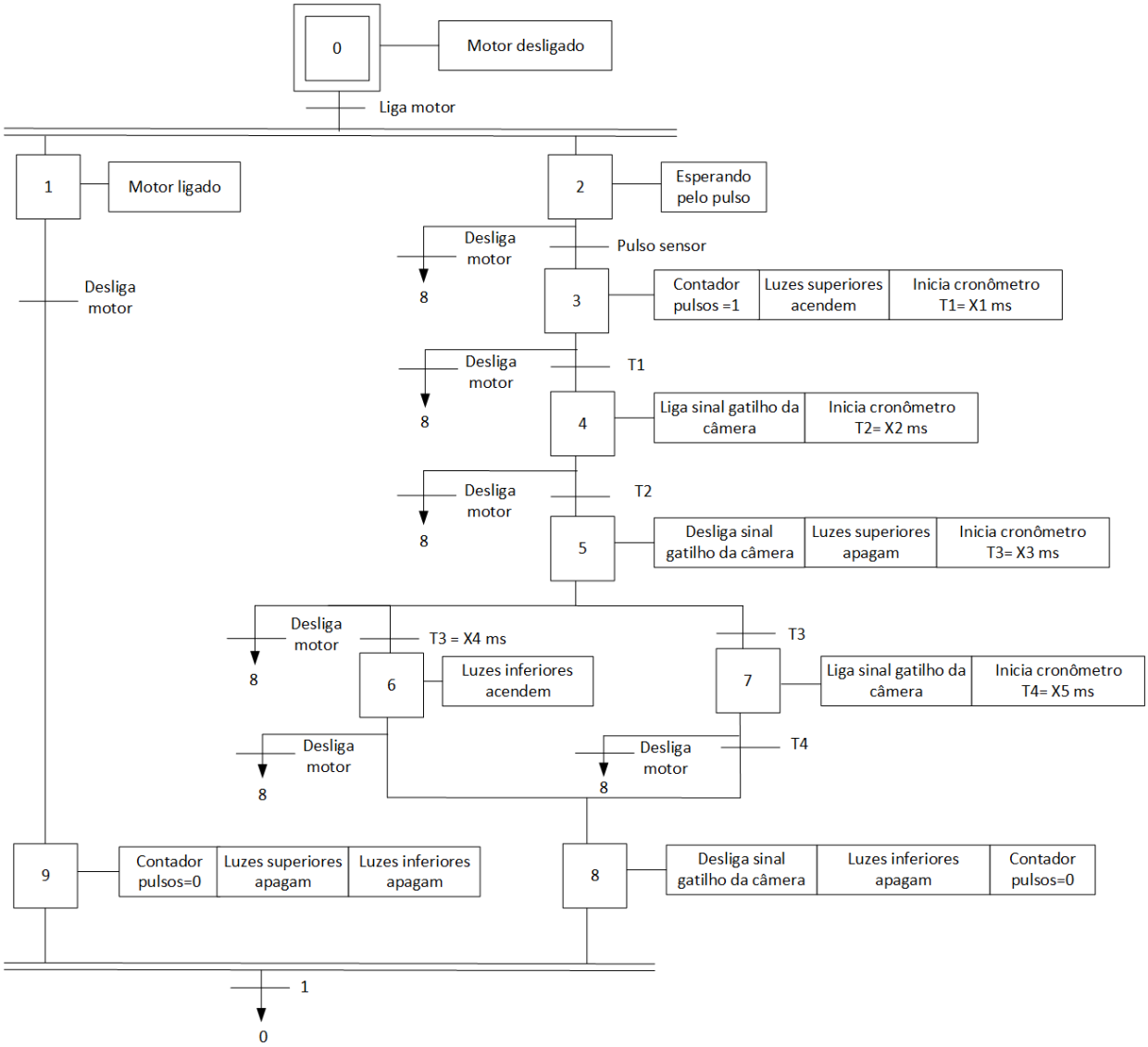


Fonte – O autor.

APÊNDICE C – CONFIGURAÇÕES DO SISTEMA PARA CAPTURA DE IMAGEM

Para a captura das imagens, o CLP foi programado de acordo com o diagrama sequencial apresentado na forma de *GRAFCET* mostrado na Figura 123, este diagrama mostra o comportamento do sistema lógico pré-definido pelas suas entradas e saídas. A primeira fase da programação exigia a determinação dos tempos necessários para garantir duas capturas da câmera (*X2, X3 e X5*) e os tempos para estabilizar a luz das lâmpadas (*X1 e X4*).

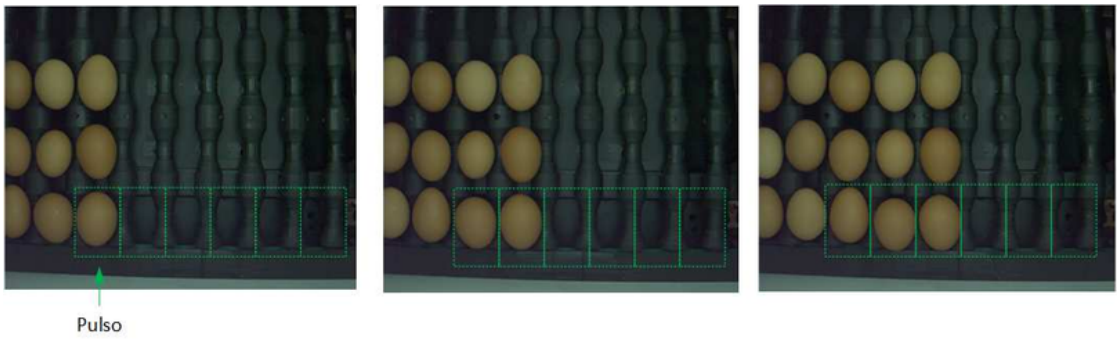
Figura 123 – Diagrama sequencial da programação do CLP.



Fonte – O autor.

As duas capturas deviam ser feitas em um intervalo máximo de 244 ms, este tempo era o tempo de chegada de um pulso a outro associado ao sensor do eixo do motor com velocidade de 60 hz. A captura neste intervalo garantiu que toda a superfície do ovo fosse capturada em 6 posições (Figura 124).

Figura 124 – Captura de imagem na chegada de cada pulso do sensor.

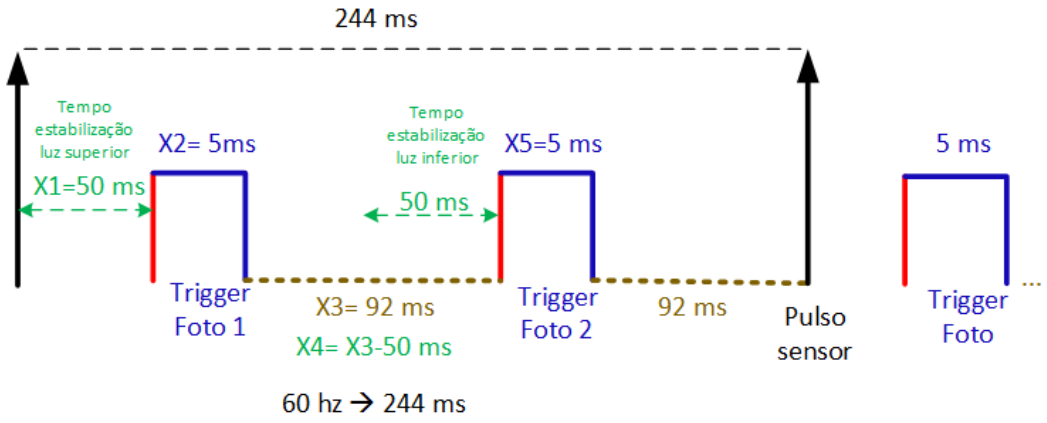


Fonte – O autor.

A câmera precisa de um tempo mínimo de disparo, o tempo mínimo encontrado é de 1 ms, contudo, para adquirir imagens de (1280 x 124) px nessa velocidade o computador deve ter uma placa de rede que aceite pacotes jumbo (*JUMBOFRAME*), caso contrário, haverá perda de dados. O tempo de disparo foi definido em 5 ms para eliminar essa possibilidade. Adicionalmente, foi estabelecido um tempo de 92 ms entre um disparo e outro para a execução dos algoritmos. Finalmente obtendo os tempos: *X2 = 5 ms, X3 = 92 ms e X5 = 5 ms.*

Uma vez determinado o tempo de captura das duas imagens, foi necessário ajustar o tempo de estabilização das lâmpadas, já que as capturas tem que ser feitas com as lâmpadas em sua potência máxima, a partir de vários testes verificou-se que os tempos mais adequados para os sistemas são os mostrados na Figura 125, conseguindo capturar uma imagem com luz superior e uma imagem com luz inferior de cada ovo.

Figura 125 – Tempos de captura das câmeras e estabilização das lâmpadas.

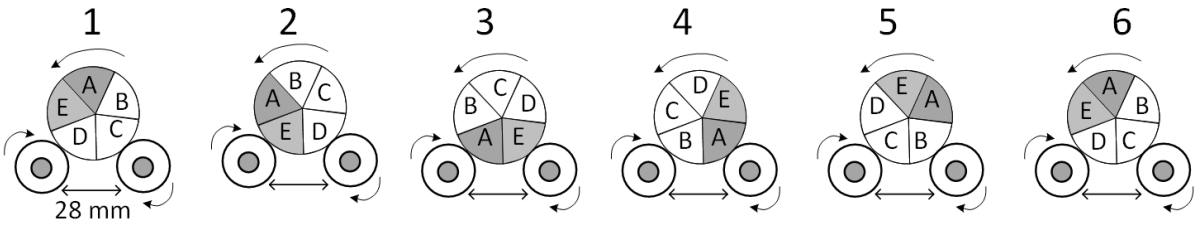


Fonte – O autor.

AQUISIÇÃO DE IMAGEM

A altura adequada para a câmera deveria garantir um campo de visão, dentro do qual cada ovo giraria 360º, permitindo o monitoramento de todas as superfícies da casca. A partir de uma série de testes verificou-se que o ovo girou 360º em uma distância total de 250 mm para um espaçamento entre os rolos de 28 mm, essa distância equivale a capturar o ovo em 5 posições, ou seja, 5 fileiras de ovos, no entanto, 6 linhas foram escolhidas para ter uma folga, conforme mostrado na Figura 126.

Figura 126 – Rotação de 360º do ovo no rolo.



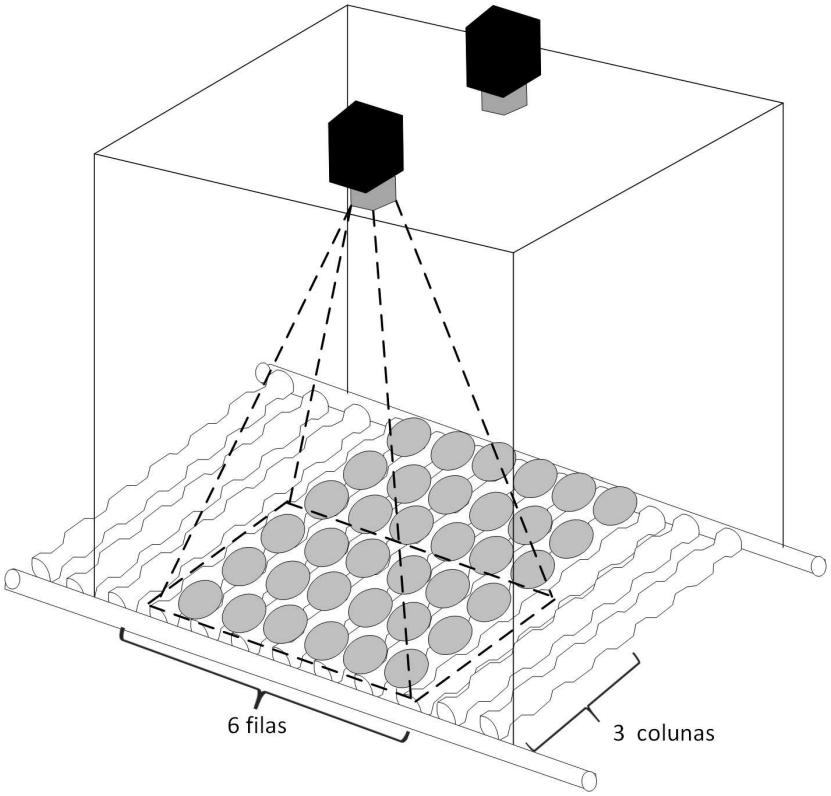
Fonte – O autor.

Para determinar o número de colunas, foi considerado o nível de detalhe da imagem associada a cada ovo individual. Levando isso em consideração, não foi possível a partir de uma única captura analisar as 6 colunas de ovos do compartimento, pois no total eram 6x6 ovos, o que fazia com que cada ovo individual perdesse resolução causando perda de detalhes, inclusive, as imagens foram afetadas pela distorção radial da câmera que fazia os ovos nas laterais da imagem parecerem arredondados conforme mostrado na Figura 129b, o que pode ser uma dificuldade na análise do formato. Para resolver este problema, foram utilizadas duas câmaras, cada uma adquirindo 6*3 ovos com uma folga nos quatro lados para cada captura conforme mostrado na Figura 128, a folga nas laterais garantiu que os ovos a serem analisados não fossem afetados pela distorção radial da câmera. Uma distância de 510 mm entre o ovo e a câmara foi selecionada para atender a esses requisitos. Um exemplo de aquisição de sistema com duas câmeras é mostrado na Figura 129.

Figura 127 – (a) Ovo no centro da câmera; (b) Ovos no lateral da câmera.

Fonte – O autor.

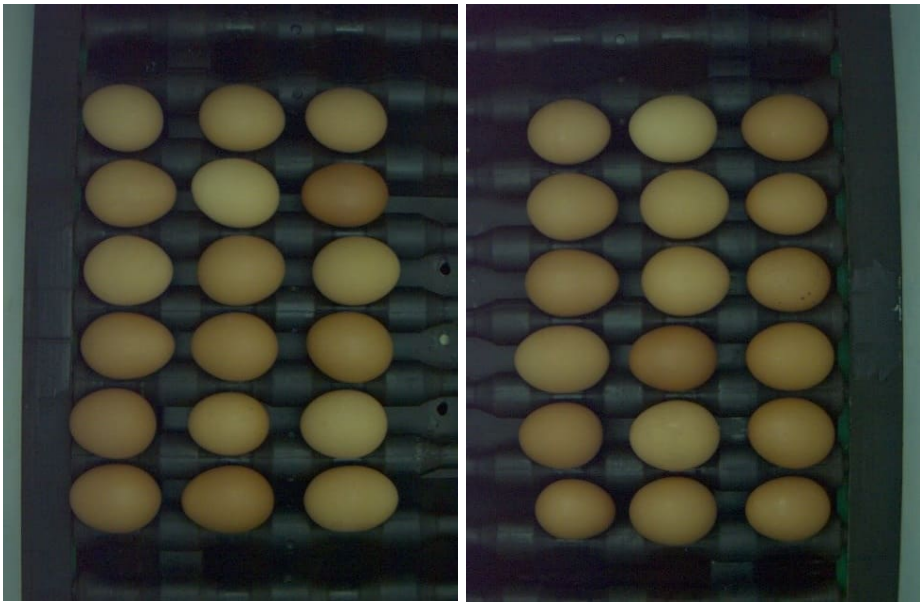
Figura 128 – Aquisição de imagens.



Fonte – O autor.

Figura 129 – Imagens obtidas pelo sistema de inspeção desenvolvido. (a) Câmera 1; (b) Câmera 2.

(a) (b)

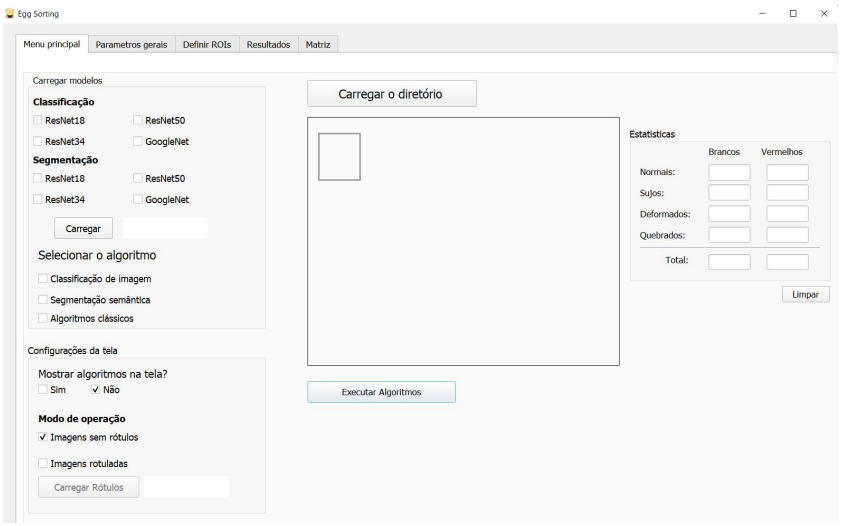


Fonte – O autor.

APÊNDICE D – INTERFACE DE USUÁRIO

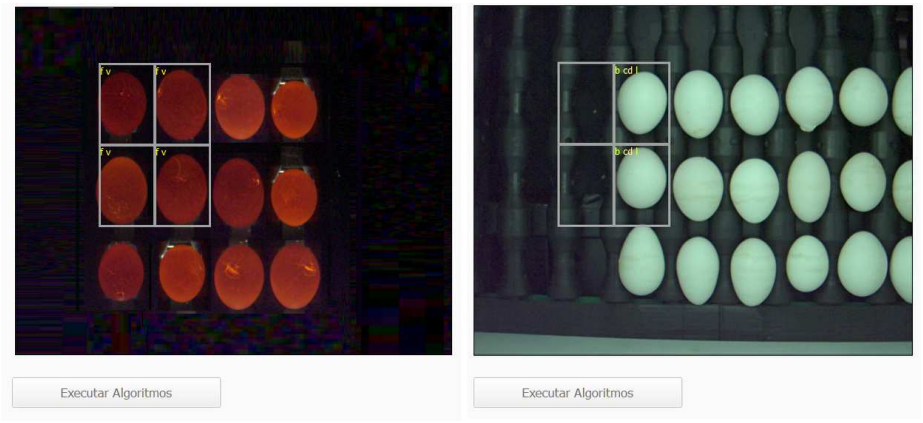
A Figura 130 mostra o menu principal da interface do usuário projetada para este projeto. A interface consiste em 5 abas: menu principal, parâmetros gerais, definição dos *ROIs*, resultados e matriz. O menu principal permite carregar os modelos e definir o algoritmo a avaliar. Também é possível configurar parâmetros adicionais como decidir se deve ou não exibir os resultados em tempo real da classificação na tela, um exemplo disso é apresentado na Figura 131.

Figura 130 – Aba Menu principal da interface do usuário desenvolvida.



Fonte – O autor.

Figura 131 – Resultados mostrados na tela em tempo real da classificação.

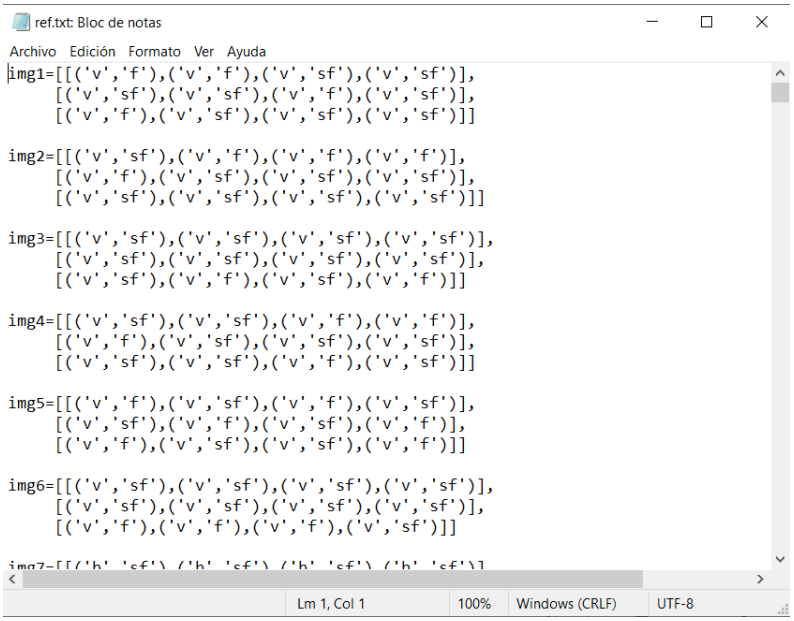


Fonte – O autor.

No menu principal também é possível definir o modo de funcionamento da interface, onde são apresentados dois modos, o modo com imagens sim rótulos e o modo com imagens com rótulos. Este último modo foi criado para calcular a precisão dos

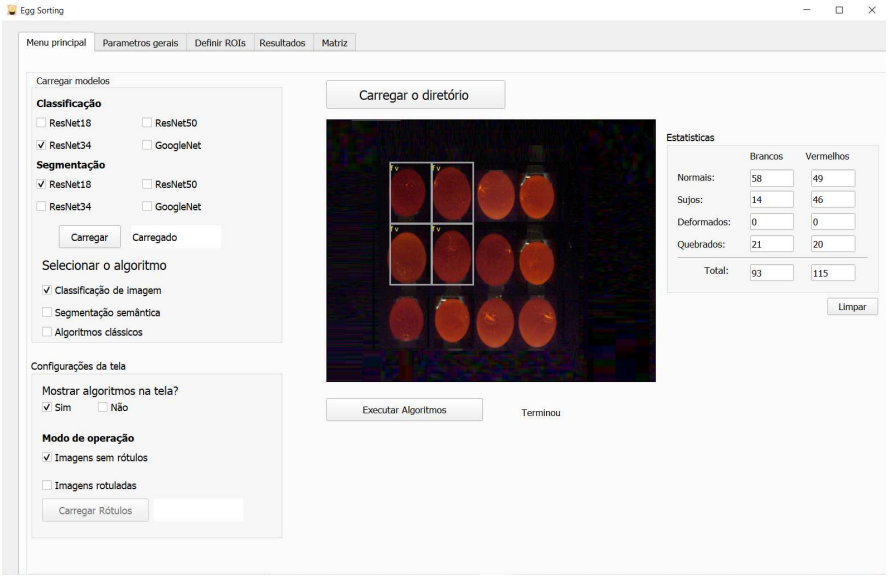
algoritmos comparando as previsões do sistema com as previsões reais das imagens. Para definir as previsões reais das imagens, foi criado um arquivo .txt mostrado na Figura 132, onde cada imagem foi associada a uma matriz com as classes de cada ovo por linha e coluna. O menu principal também permite visualizar as estatísticas do total de ovos brancos e vermelhos classificados como normais, ou seja, limpos, sem deformidades e sem fissuras, sujos com pelo menos um grau de sujeira, deformados e quebrados (Figura 133).

Figura 132 – Arquivo .txt que armazena os rótulos reais das imagens.



Fonte – O autor.

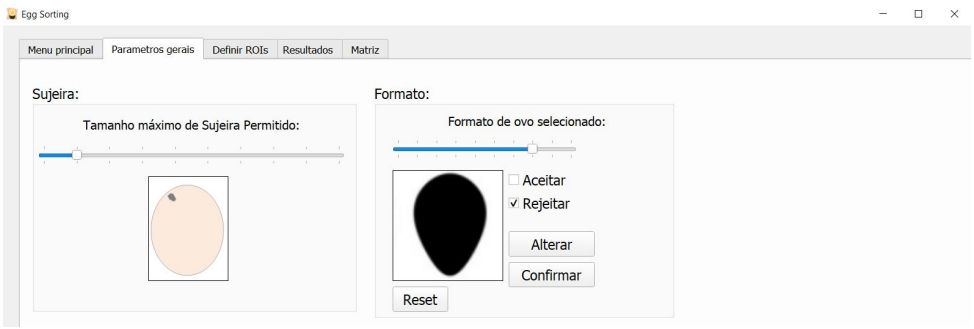
Figura 133 – Estatísticas do total de ovos classificados.



Fonte – O autor.

Na aba “parâmetros gerais” (Figura 134) é possível configurar o tamanho máximo de sujeira permitido no caso de algoritmos clássicos e o formato do ovo considerado aceitável ou descartável.

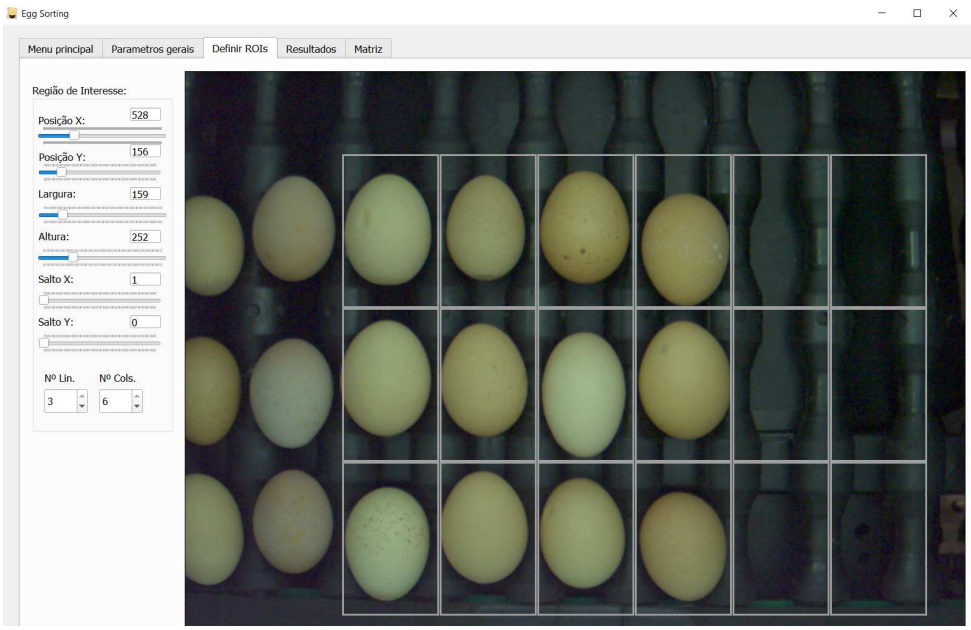
Figura 134 – Configuração dos parâmetros gerais dos algoritmos na interface.



Fonte – O autor.

Na aba “Definir Rois” (Figura 135) é possível localizar as posições dos ovos a serem analisados onde há no máximo 6 colunas e 3 linhas. Nesta interface é possível modificar a largura, a altura e a distância entre as regiões de interesse.

Figura 135 – Localização das regiões de interesse na interface.

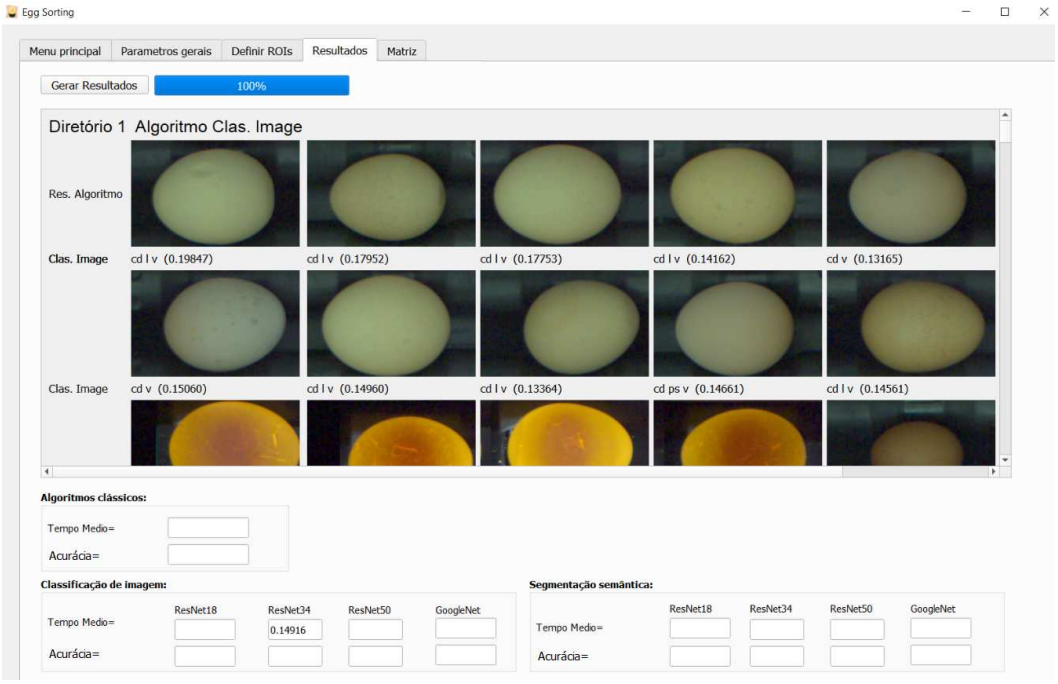


Fonte – O autor.

Na aba “Resultados” quando executado no modo imagens sim rótulos (Figura 136), é apresentado apenas o resultado previsto pelos algoritmos com seus respectivos tempos de processamento por imagem. No caso de trabalhar com imagens rotuladas, a comparação do resultado do rótulo real é apresentada juntamente com o rótulo predito

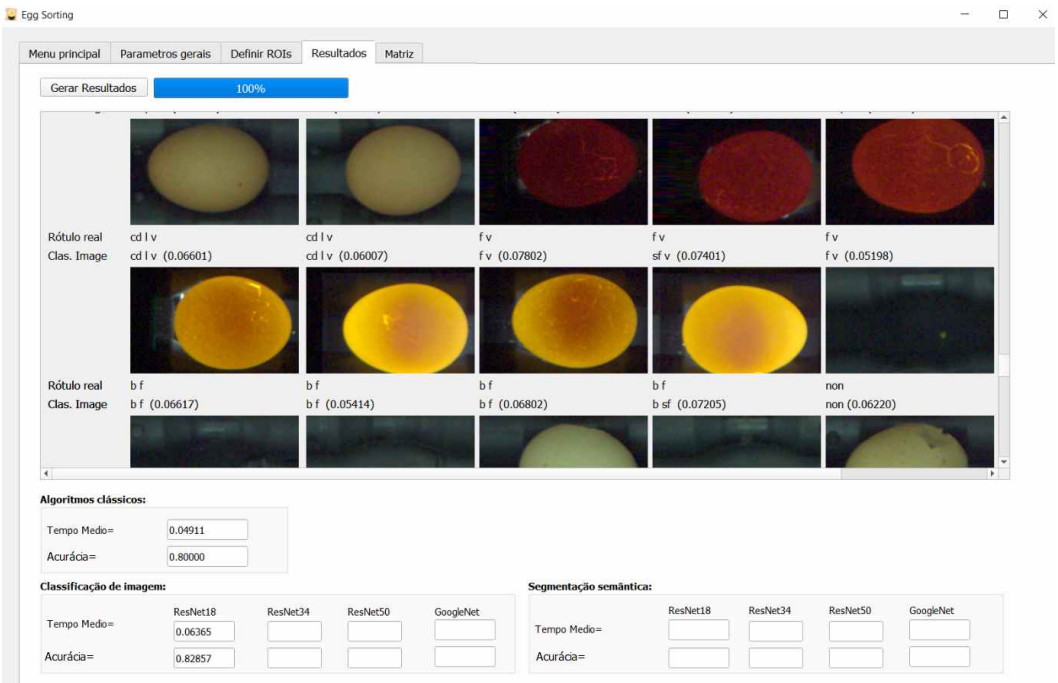
pelo modelo, ao final são apresentadas as estatísticas médias obtidas pelos algoritmos (Figura 137).

Figura 136 – Resultados classificação modo imagens sim rótulos.



Fonte – O autor.

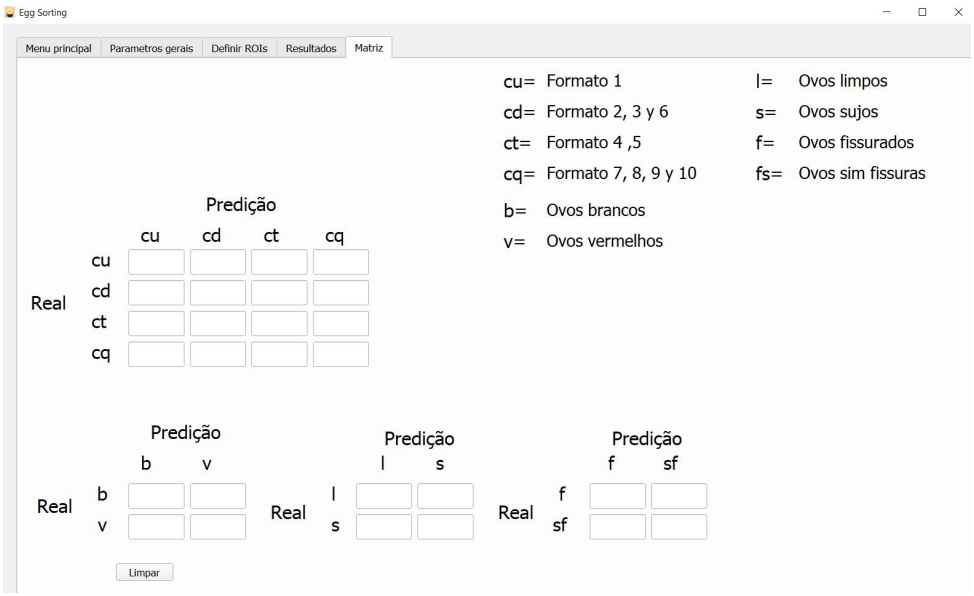
Figura 137 – Resultados classificação modo imagens com rótulos.



Fonte – O autor.

Finalmente, a aba “Matriz” (Figura 138) permite visualizar os resultados obtidos em uma matriz de confusão para cada categoria.

Figura 138 – Matriz de confusão para visualizar os resultados.



Fonte – O autor.

APÊNDICE E – MATRIZES DE CONFUSÃO E MÉTRICAS DE AVALIAÇÃO

Este apêndice apresenta as matrizes de confusão e as métricas de avaliação dos resultados obtidos pelos modelos de aprendizagem profundo na classificação de imagens e na classificação usando segmentação semântica.

Tabela 26 – Matriz de confusão e métricas de avaliação dos resultados obtidos na classificação de imagens com *Resnet18*.

		Predição																	
	Classes	**cu**	**cd**	**ct**	**cq**	**b_LS**	**v_LS**	**b_LI**	**v_LI**	**ps**	**ms**	**l**	**f**	**sf**	**non**	**ND**	**FN**		***Precisão***	***Recall***	***F1-Score***
Real	**cu**	**31**	16													3	19		0.816	0.620	0.705
	cd	7	**982**		2											14	23		0.928	0.977	0.952
	ct		34	**10**	10											21	65		0.769	0.133	0.227
	cq		26	3	**39**											21	50		0.765	0.438	0.557
	b_LS					**743**										0	0		0.999	1.000	0.999
	v_LS					1	**479**									0	1		1	0.998	0.999
	b_LI							**119**	1							0	1		1	0.992	0.996
	v_LI								**128**							0	0		0.992	1	0.996
	ps									**335**	16	56				52	124		0.792	0.730	0.760
	ms									29	**100**					16	45		0.862	0.690	0.766
	l									59		**532**				30	89		0.905	0.857	0.880
	f												**159**	5		1	6		0.874	0.964	0.916
	sf												23	**52**		8	31		0.912	0.627	0.743
	non														**369**	0	0		1	1	1
	FP	7	76	3	12	1	0	0	1	88	16	56	23	5	0	166		**Média**	0.900	0.793	0.825

Notas: LS: Luz superior
LI: Luz inferior
ND: No detectado
FP: Falsos positivos
FN: Falsos negativos

Tabela 27 – Matriz de confusão e métricas de avaliação dos resultados obtidos na classificação de imagens com *ResNet34*.

		Predição																	
	Classes	**cu**	**cd**	**ct**	**cq**	**b_LS**	**v_LS**	**b_LI**	**v_LI**	**ps**	**ms**	**l**	**f**	**sf**	**non**	**ND**	**FN**		***Precisão***	***Recall***	***F1-Score***
Real	**cu**	**39**	8													3	11		0.78	0.78	0.78
	cd	11	**983**	3												8	22		0.956	0.978	0.967
	ct		27	**28**	7											13	47		0.824	0.373	0.514
	cq		10	3	**71**											5	18		0.910	0.798	0.850
	b_LS					**743**										0	0		1	1	1
	v_LS						**480**									0	0		1	1	1
	b_LI							**120**								0	0		1	1	1
	v_LI								**128**							0	0		1	1	1
	ps									**373**	27	45				14	86		0.846	0.813	0.829
	ms									18	**124**					3	21		0.821	0.855	0.838
	l									50		**559**				12	62		0.925	0.900	0.913
	f												**149**	10		6	16		1	0.903	0.949
	sf													**82**		0	1		0.891	0.988	0.937
	non														**369**	0	0		1	1	1
	FP	11	45	6	7	0	0	0	0	68	27	45	0	10	0	143		**Média**	0.912	0.877	0.888

Notas: LS: Luz superior
LI: Luz inferior
ND: No detectado
FP: Falsos positivos
FN: Falsos negativos

Tabela 28 – Matriz de confusão e métricas de avaliação dos resultados obtidos na classificação de imagens com *ResNet50*.

		Predição																		
	Classes	**cu**	**cd**	**ct**	**cq**	**b_LS**	**v_LS**	**b_LI**	**v_LI**	**ps**	**ms**	**l**	**f**	**sf**	**non**	**ND**	**FN**		***Precisão***	***Recall***	***F1-Score***
Real	**cu**	**43**	5													2	7		0.860	0.860	0.860
	cd	7	**990**	1												7	15		0.972	0.985	0.979
	ct		17	**38**	7											13	37		0.927	0.507	0.655
	cq		6	2	**76**											5	13		0.916	0.854	0.884
	b_LS					**743**										0	0		1	1	1
	v_LS						**480**									0	0		1	1	1
	b_LI							**119**	1							0	1		1	0.992	0.996
	v_LI								**128**							0	0		0.992	1	0.996
	ps									**392**	25	20				22	67		0.942	0.854	0.896
	ms									12	**133**					0	12		0.842	0.917	0.878
	l									12		**598**				11	23		0.968	0.963	0.965
	f												**144**	17		4	21		1	0.873	0.932
	sf													**82**		0	1		0.828	0.988	0.901
	non														**369**	0	0		1	1	1
	FP	7	28	3	7	0	0	0	1	24	25	20	0	17	0	101		**Média**	0.935	0.911	0.918

Notas: LS: Luz superior
LI: Luz inferior
ND: No detectado
FP: Falsos positivos
FN: Falsos negativos

Tabela 29 – Matriz de confusão e métricas de avaliação dos resultados obtidos na classificação de imagens com *GoogLeNet*.

		Predição																		
	Classes	**cu**	**cd**	**ct**	**cq**	**b_LS**	**v_LS**	**b_LI**	**v_LI**	**ps**	**ms**	**l**	**f**	**sf**	**non**	**ND**	**FN**		***Precisão***	***Recall***	***F1-Score***
Real	**cu**	**35**	14													1	15		0.897	0.700	0.787
	cd	4	**1000**													1	5		0.934	0.995	0.963
	ct		39	**13**	6											17	62		0.867	0.173	0.289
	cq		18	2	**60**											9	29		0.909	0.674	0.774
	b_LS					**743**										0	0		0.999	1	0.999
	v_LS					1	**479**									0	1		1	0.998	0.999
	b_LI							**114**	5							1	6		1	0.950	0.974
	v_LI								**128**							0	0		0.962	1	0.981
	ps									**296**	25	104				34	163		0.868	0.645	0.740
	ms									19	**124**					2	21		0.832	0.855	0.844
	l									26		**585**				10	36		0.849	0.942	0.893
	f												**157**	6		2	8		0.902	0.952	0.926
	sf												17	**63**		3	20		0.913	0.759	0.829
	non														**369**	0	0		1	1	1
	FP	4	71	2	6	1	0	0	5	45	25	104	17	6	0	177		**Média**	0.928	0.832	0.861

Notas: LS: Luz superior
LI: Luz inferior
ND: No detectado
FP: Falsos positivos
FN: Falsos negativos

Tabela 30 – Matriz de confusão e métricas de avaliação dos resultados obtidos na classificação usando segmentação semântica com *ResNet18*.

		Predição																		
	Classes	**cu**	**cd**	**ct**	**cq**	**b_LS**	**v_LS**	**b_LI**	**v_LI**	**ps**	**ms**	**l**	**f**	**sf**	**non**	**FN**		***Precisão***	***Recall***	***F1-Score***
Real	**cu**	**30**	14		6											20		0.429	0.600	0.500
	cd	40	**901**	14	50											104		0.971	0.897	0.932
	ct		10	**55**	10											20		0.534	0.733	0.618
	cq		3	34	**52**											37		0.441	0.584	0.502
	b_LS					**743**										0		1	1	1
	v_LS						**480**									0		1	1	1
	b_LI							**120**								0		1	1	1
	v_LI								**128**							0		1	1	1
	ps									**269**	117	92				209		0.659	0.563	0.607
	ms									40	**101**	4				44		0.461	0.697	0.555
	l									99	1	**498**				100		0.838	0.833	0.836
	f												**158**	7		7		0.958	0.958	0.958
	sf												7	**76**		7		0.916	0.916	0.916
	non														**369**	0		1	1	1
	FP	40	27	48	66	0	0	0	0	139	118	96	7	7	0	0		**Média**	0.797	0.823	0.804

Notas: LS: Luz superior
LI: Luz inferior
FP: Falsos positivos
FN: Falsos negativos

Tabela 31 – Matriz de confusão e métricas de avaliação dos resultados obtidos na classificação usando segmentação semântica com *ResNet34*.

		Predição																	
	Classes	**cu**	**cd**	**ct**	**cq**	**b_LS**	**v_LS**	**b_LI**	**v_LI**	**ps**	**ms**	**l**	**f**	**sf**	**non**	**FN**		***Precisão***	***Recall***	***F1-Score***
Real	**cu**	**30**	13		7											20		0.469	0.600	0.526
	cd	34	**935**	16	20											70		0.964	0.930	0.947
	ct		16	**58**	1											17		0.532	0.773	0.630
	cq		6	35	**48**											41		0.632	0.539	0.582
	b_LS					**743**										0		1	1	1
	v_LS						**480**									0		1	1	1
	b_LI							**120**								0		1	1	1
	v_LI								**128**							0		1	1	1
	ps									**256**	111	111				222		0.738	0.536	0.621
	ms									41	**102**	2				43		0.479	0.703	0.570
	l									50		**548**				50		0.829	0.916	0.871
	f												**164**	1		1		1.000	0.994	0.997
	sf													**79**		4		0.988	0.952	0.969
	non														**369**	0		1	1	1
	FP	34	35	51	28	0	0	0	0	91	111	113	0	1	0	0	**Média**	0.822	0.834	0.822

Notas: LS: Luz superior
LI: Luz inferior
FP: Falsos positivos
FN: Falsos negativos

Tabela 32 – Matriz de confusão e métricas de avaliação dos resultados obtidos na classificação usando segmentação semântica com *ResNet50*.

		Predição																	
	Classes	**cu**	**cd**	**ct**	**cq**	**b_LS**	**v_LS**	**b_LI**	**v_LI**	**ps**	**ms**	**l**	**f**	**sf**	**non**	**FN**		***Precisão***	***Recall***	***F1-Score***
Real	**cu**	**30**	12		8											20		0.469	0.600	0.526
	cd	34	**933**	14	24											72		0.967	0.928	0.947
	ct		15	**57**	3											18		0.500	0.760	0.603
	cq		5	43	**41**											48		0.539	0.461	0.497
	b_LS					**743**										0		1	1	1
	v_LS						**480**									0		1	1	1
	b_LI							**120**								0		1	1	1
	v_LI								**128**							0		1	1	1
	ps									**260**	117	101				218		0.712	0.544	0.617
	ms									33	**103**	9				42		0.460	0.710	0.558
	l									72	4	**522**				76		0.826	0.873	0.849
	f												**164**	1		1		0.982	0.994	0.988
	sf												3	**80**		3		0.988	0.964	0.976
	non														**369**	0		1	1	1
	FP	34	32	57	35	0	0	0	0	105	121	110	3	1	0		**Média**	0.809	0.821	0.807

Notas: LS: Luz superior
LI: Luz inferior
FP: Falsos positivos
FN: Falsos negativos

Tabela 33 – Matriz de confusão e métricas de avaliação dos resultados obtidos na classificação usando segmentação semântica com *GoogLeNet*.

		Predição																	
	Classes	**cu**	**cd**	**ct**	**cq**	**b_LS**	**v_LS**	**b_LI**	**v_LI**	**ps**	**ms**	**l**	**f**	**sf**	**non**	**FN**		***Precisão***	***Recall***	***F1-Score***
Real	**cu**	**30**	12		8											20		0.508	0.600	0.550
	cd	29	**919**	7	50											86		0.971	0.914	0.942
	ct		12	**60**	3											15		0.588	0.800	0.678
	cq		3	35	**51**											38		0.455	0.573	0.507
	b_LS					**743**										0		1	1	1
	v_LS						**480**									0		1	1	1
	b_LI							**120**								0		1	1	1
	v_LI								**128**							0		1	1	1
	ps									**250**	147	81				228		0.653	0.523	0.581
	ms									37	**99**	9				46		0.401	0.683	0.505
	l									96	1	**501**				97		0.848	0.838	0.843
	f												**163**	2		2		0.976	0.988	0.982
	sf												4	**79**		4		0.975	0.952	0.963
	non														**369**	0		1	1	1
	FP	29	27	42	61	0	0	0	0	133	148	90	4	2	0		**Média**	0.792	0.823	0.803

Notas: LS: Luz superior
LI: Luz inferior
FP: Falsos positivos
FN: Falsos negativos

—