UNIVERSIDADE ESTADUAL PAULISTA
"JULIO DE MESQUITA FILHO"
INSTITUTO DE GEOCIÊNCIAS E CIÊNCIAS EXATAS

Trabalho de Conclusão de Curso

Curso de Graduação em Física

DETECÇÃO E CLASSIFICAÇÃO DE GEMAS DE CANA-DE-AÇÚCAR POR VISÃO COMPUTACIONAL: UMA ABORDAGEM BASEADA EM REDES NEURAIS CONVOLUCIONAIS

Enzo Carlini Giovannetti

Prof. Dr. Alexandre Mesquita

Rio Claro (SP)

2025

UNIVERSIDADE ESTADUAL PAULISTA

Instituto de Geociências e Ciências Exatas

Campus de Rio Claro

ENZO CARLINI GIOVANNETTI

DETECÇÃO E CLASSIFICAÇÃO DE GEMAS DE CANA-DE-AÇÚCAR POR VISÃO COMPUTACIONAL: UMA ABORDAGEM BASEADA EM REDES NEURAIS CONVOLUCIONAIS

Trabalho de Conclusão de Curso apresentado ao Instituto de Geociências e Ciências Exatas - Câmpus de Rio Claro, da Universidade Estadual Paulista Júlio de Mesquita Filho, para obtenção do grau de Bacharel em Física.

Rio Claro - SP

2025

G512d

Giovannetti, Enzo Carlini

Detecção e classificação de gemas de cana-de-açúcar por visão computacional: uma abordagem baseada em redes neurais convolucionais / Enzo Carlini Giovannetti. -- Rio Claro, 2025

36 p. : il.

Trabalho de conclusão de curso (Bacharelado - Física) - Universidade Estadual Paulista (UNESP), Instituto de Geociências e Ciências Exatas, Rio Claro

Orientador: Alexandre Mesquita

1. Introdução. 2. Física das redes neurais. 3. Visão Computacional e Reconhecimento de Padrões. 4. Metodologia. 5. Resultados. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Dados fornecidos pelo autor(a).

ENZO CARLINI GIOVANNETTI

DETECÇÃO E CLASSIFICAÇÃO DE GEMAS DE CANA-DE-AÇÚCAR POR VISÃO COMPUTACIONAL: UMA ABORDAGEM BASEADA EM REDES NEURAIS CONVOLUCIONAIS

Trabalho de Conclusão de Curso apresentado ao Instituto de Geociências e Ciências Exatas - Câmpus de Rio Claro, da Universidade Estadual Paulista Júlio de Mesquita Filho, para obtenção do grau de Bacharel em Física.

Comissão Examinadora

Alexandre Mesquita

Ricardo Paupitz Barbosa dos Santos

Adriano José Galvani Otuka

Rio Claro, 18 de Novembro de 2025

gov.br
Documento assinado digitalmente
ENZO CARLINI GIOVANNETTI
Data: 02/12/2025 11:31:58-0300
Verifique em https://validar.iti.gov.br

Assinatura do aluno

unesp
UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"

DN: cn=Alexandre Mesquita:22240210893, ou=UNESP - Universidade Estadual Paulista Julio de Mesquita Filho, o=ICPEdu, c=BR
Localização: https://pessoal.icpedu.rnp.br/public/verificar-assinatura
Dados: 2025.12.02 09:45:54 -03'0

Assinatura do orientador

AGRADECIMENTOS

Agradeço ao meu orientador, Alexandre Mesquita, que me apoiou, guiou e me ensinou em todas as etapas e degraus ao longo da minha graduação.

Agradeço aos membros da banca por me darem essa oportunidade de aprendizado e transformação de conhecimento, junto com um agradecimento à faculdade UNESP, ao campus de Rio Claro, e ao Instituto de Física.

Agradeço aos meus pais: à minha mãe, Adriana Carlini, que me ensinou a aprender, a ouvir; e ao meu pai, Erico Giovannetti, que me ensinou disciplina e respeito. A ambos, agradeço por me ensinarem a ser gentil e esperançoso, e por sempre me lembrarem que o melhor ainda está por vir.

Agradeço aos meus amigos, Matheus Calil, Thabata Ismail, Thiago Chaim e Camila Ferro, que estiveram literalmente ao meu lado nesta jornada inteira, me apoiando em todas as esquinas.

Ainda mais, serei eternamente grato à minha amiga do coração, minha companheira, Camila Ferro, que nunca duvidou de mim. Nunca esquecerei sua bondade e compaixão.

RESUMO

No contexto da modernização agrícola, este trabalho estuda a aplicação de técnicas de reconhecimento visual para identificar gemas de cana-de-açúcar, utilizando redes neurais artificiais no ambiente MATLAB. O objetivo central é distinguir entre gemas adequadas e inadequadas para o replantio, visando contribuir para a automatização de um processo atualmente manual. Para tanto, foram inicialmente avaliados métodos tradicionais de processamento de imagens, como reconhecimento de padrões e análise de diferenças de fundo, que, apesar de úteis como ponto de partida, mostraram limitações em precisão e robustez. Em seguida, aplicaram-se arquiteturas de redes neurais profundas (GoogleNet, Darknet e YOLO) disponíveis no MATLAB. O conjunto de dados, composto por imagens de campo obtidas próximo a Rio Claro, contou com aproximadamente 20 gemas inviáveis e 15 viáveis para treinamento e validação. Do ponto de vista físico, a pesquisa insere-se na análise quantitativa de imagens para caracterização de estruturas naturais, explorando conceitos de óptica e processamento de informação. A expectativa é que o uso de redes neurais, mesmo com uma base de dados reduzida, permita uma classificação inicial satisfatória, demonstrando a viabilidade da abordagem e apontando para futuras expansões, como o aumento do banco de imagens e a otimização de arquiteturas, alinhando-se assim ao desenvolvimento de técnicas de instrumentação com embasamento físico.

Palavras-chave: redes neurais; reconhecimento de imagens; cana-de-açúcar; visão computacional; automatização agrícola.

ABSTRACT

In the context of agricultural modernization, this work investigates the application of visual recognition techniques to identify sugarcane buds using artificial neural networks in the MATLAB environment. The main objective is to distinguish between suitable and unsuitable buds for replanting, aiming to contribute to the automation of a process that is currently manual. To this end, traditional image-processing methods were initially evaluated, such as pattern recognition and background-difference analysis, which, although useful as a starting point, showed limitations in accuracy and robustness. Subsequently, deep neural-network architectures available in MATLAB (GoogleNet, Darknet, and YOLO) were applied. The dataset, consisting of field images collected near Rio Claro, included approximately 20 nonviable and 15 viable buds for training and validation. From a physical standpoint, the research is situated within the quantitative analysis of images for the characterization of natural structures, exploring concepts of optics and information processing. The expectation is that the use of neural networks, even with a reduced dataset, will enable a satisfactory initial classification, demonstrating the feasibility of the approach and pointing toward future expansions, such as enlarging the image repository and optimizing architectures, thus aligning with the development of instrumentation techniques grounded in physics.

Keywords: neural networks; image recognition; sugarcane; computer vision; agricultural automation.

Sumário

1. Introdução

As redes neurais artificiais representam uma das mais importantes ferramentas contemporâneas no campo da inteligência artificial. Inspiradas na estrutura biológica do cérebro humano, elas operam por meio de unidades matemáticas denominadas neurônios artificiais, os quais se organizam em camadas sucessivas. Cada neurônio recebe entradas, pondera tais sinais mediante pesos ajustáveis e, em seguida, aplica uma função de ativação, cujo papel é introduzir não linearidade ao sistema. É essa não linearidade que possibilita a resolução de problemas complexos, como o reconhecimento de imagens e padrões de alta dimensionalidade. O processo de aprendizado ocorre através de ajustes nos pesos sinápticos, utilizando-se algoritmos de otimização, dos quais o mais comum é o chamado gradiente descendente. Dessa forma, redes neurais podem, progressivamente, minimizar erros na tarefa para a qual foram projetadas. Ainda que a ideia remonte às décadas de 1940 e 1950, apenas com o advento da computação de alta performance, do armazenamento massivo de dados e das arquiteturas profundas é que se tornaram plenamente aplicáveis em escala industrial e científica.(1,2,3,4)

No contexto agrícola, como no caso do reconhecimento óptico de gemas de cana-de-açúcar, o funcionamento básico das redes neurais mostra-se particularmente adequado. Isso porque a tarefa envolve distinguir pequenas variações visuais, muitas vezes imperceptíveis ao olho humano, mas passíveis de serem identificadas por modelos adequadamente treinados.(5,6,7)

As arquiteturas modernas de redes neurais distinguem-se pela profundidade, isto é, pela quantidade de camadas entre a entrada e a saída do sistema. Em redes rasas, compostas por poucas camadas, há limitação quanto à capacidade de abstração. Já as chamadas redes profundas — ou deep neural networks — conseguem extrair representações hierárquicas cada vez mais sofisticadas dos dados de entrada. Na prática, isso significa que as primeiras camadas de uma rede profunda dedicada à visão computacional podem aprender a identificar formas simples, como arestas e texturas. Camadas intermediárias passam a reconhecer estruturas mais complexas, como contornos ou regiões específicas da imagem. Finalmente, as últimas camadas são responsáveis por composições ainda mais abstratas, chegando à classificação ou detecção do objeto de interesse. Esse caráter hierárquico é fundamental para aplicações na área agrícola, uma vez que a classificação de gemas de cana-de-açúcar envolve detalhes de textura e morfologia. Uma rede rasa dificilmente distinguiria, com acurácia suficiente, uma gema saudável de uma inviável para replantio. Entretanto, uma rede profunda, devido à sua

capacidade de representação multiescala, pode alcançar resultados muito superiores, aproximando-se da percepção visual humana, mas com maior objetividade e consistência.(3,8,1,7)

Nos últimos anos, verificaram-se avanços significativos no aprendizado profundo (deep learning), destacando-se não apenas pela ampliação da capacidade computacional, mas também por inovações conceituais. Entre elas, ressaltam-se arquiteturas como as redes convolucionais (Convolutional Neural Networks – CNNs), essenciais para tarefas de visão computacional, e as redes transformadoras (transformers), que revolucionaram o processamento de linguagem natural, mas vêm sendo progressivamente adaptadas para aplicações visuais. As CNNs, por exemplo, operam mediante filtros convolucionais que percorrem a imagem, extraindo automaticamente padrões relevantes, sem a necessidade de intervenção humana no processo de definição de características. Tal abordagem tem se mostrado particularmente eficiente em problemas como o reconhecimento de imagens médicas, análise de satélites e, no presente caso, no diagnóstico automatizado da qualidade de gemas agrícolas.(2,3,8,9,5)

Um marco na valorização acadêmica e científica desse campo foi o Prêmio Nobel de Física de 2024, concedido a pesquisadores cujas contribuições reforçam a conexão entre teoria física e aprendizado de máquina (ver nota). Este reconhecimento ilustra a relevância global do tema, indicando que o impacto das redes neurais transcende áreas aplicadas e alcança questões fundamentais sobre a representação da informação e a dinâmica de sistemas complexos.(10,11,12)

A cultura da cana-de-açúcar, de extrema importância para o agronegócio brasileiro, enfrenta desafios significativos em sua fase de replantio. O processo tradicional de multiplicação vegetativa exige a seleção e o corte manual de colmos (gemas) em viveiros, uma operação que consome grande quantidade de mão de obra, é intrinsecamente perigosa devido à manipulação de ferramentas afiadas e está sujeita a uma alta variabilidade subjetiva na classificação da qualidade das gemas. Esta ineficiência representa um gargalo operacional e econômico, criando uma demanda clara por soluções automatizadas que possam oferecer ganhos em segurança, velocidade e padronização. O presente trabalho tem como objetivo central investigar e desenvolver um sistema automatizado de reconhecimento visual para a distinção de gemas de cana-de-açúcar, classificando-as como "boas" ou "ruins" para replantio. Para tal, propõe-se a implementação de uma arquitetura de rede neural convolucional profunda do tipo YOLO (You Only Look Once), versão 2, capaz de realizar a detecção e a classificação localizada dos entrenós em imagens reais dos colmos.(13,5,6,14,15)

A validação do sistema será realizada por meio de métricas de precisão, recall e análise da curva Precision-Recall, confrontando suas previsões com um conjunto de dados anotado manualmente (ground truth). A hipótese a ser testada é a de que uma CNN como a YOLOv2, mesmo treinada com um conjunto de dados limitado, é capaz de aprender características visuais suficientemente discriminativas — relacionadas à textura, cor e integridade morfológica — para executar esta tarefa com um nível de acurácia que comprove a viabilidade da automação.(16,17,18,19)

A conclusão bem-sucedida desta pesquisa visa fornecer uma prova de conceito sólida para a substituição do método manual de seleção, pavimentando o caminho para o desenvolvimento de equipamentos agrícolas inteligentes e mais seguros.

2. Fisica das redes neurais.

2.1 Neurônios, microestados e energia efetiva

Sistema composto por muitas variáveis microscópicas (as ativações dos neurônios) que interagem entre si. Em modelos clássicos de inspiração física, como as redes de Hopfield, cada neurônio é tratado como uma variável discreta $s_i \in \{-1, +1\}$ ou contínua e o comportamento coletivo é descrito por uma função energia $E(\{s_i\})$. Um exemplo simples de energia de Hopfield é:

$$E(s) = \frac{1}{2}\sum_{i \neq j} J_{ij} S_i S_j - \sum_i h_i s_i . \quad (1)$$

Onde J_{ij} são as interações (pesos) entre as unidades e h_i termos de campo (vieses). Estados de baixo E correspondem a padrões memoráveis (atratores), enquanto estados de energia mais alta são instáveis. Essa formulação faz a conexão direta entre redes neurais e modelos de spins em matéria condensada, permitindo o uso de técnicas da física estatística para analisar capacidade de memória, estabilidade e transições de fase do sistema. As ideias originadas por Hopfield e a relação explícita com funções energia e distribuições do tipo Boltzmann foram fundamentais para estabelecer um arcabouço físico para modelos de aprendizado.(11,20,21)

Matematicamente, a "passagem" de informação de uma unidade $j$para outra ié descrita pelo campo local (ou campo total) que atua sobre i:(22)

$$h_i^{loc} = \sum_j J_{ij} S_j + h_i. \quad (2)$$

A interpretação probabilística complementa essa visão: o conjunto de configurações S pode ser associado a uma distribuição canônica do tipo Boltzmann:

$$P(s) = \frac{1}{Z} e^{\beta E(s)}. \quad (3)$$

Com Z a partiçao de $\beta = (\frac{1}{k_B T})$ uma temperatura efetiva que regula flutuaçoes. Em máquinas estocásticas (ex.: Boltzmann machines), o treinamento busca ajustar Jij de modo que a distribuição do modelo aproxime a distribuição empírica dos dados — um problema análogo a identificar Hamiltonianos que reproduzem a estatística observada em física.(12,21)

Em modelos determinísticos uma regra de atualização simples é $s_i = \text{sign}\ (h_i^{\text{loc}})$, enquanto em modelos estocásticos (às máquinas de Boltzmann) a ativação segue uma probabilidade dependente do campo local, por exemplo:

$$P(s_i = +1) = \sigma(\beta h_i^{loc}) = \frac{1}{1+e^{-2\beta h_i^{loc}}}, \quad (4)$$

o que conecta diretamente a dinâmica local às distribuições canônicas discutidas acima. Em arquiteturas em camadas (redes feedforward), essa passagem é análoga à aplicação sucessiva de transformações lineares seguidas de não linearidades: $a^{(l)} = \phi(W^{(l)}a^{(l-1)} + b^{(l)})$. Assim, sinais podem "pular" entre unidades e atravessar camadas por caminhos compostos de pesos — matematicamente, circuitos de conexões produzem acoplamentos efetivos entre unidades distantes que são produtos e somas dos pesos ao longo dos caminhos, e esses acoplamentos efetivos moldam quais componentes da informação são amplificados ou atenuados.(22)

2.2 Camadas, coarse-graining e emergência de representações

A arquitetura em camadas das redes profundas pode ser compreendida, por analogia, como uma sequência de operações de coarse-graining (agrupamento grosseiro). Cada camada transforma a representação dos dados, filtrando detalhes de alta frequência e extraindo as variáveis de ordem — isto é, os elementos mais relevantes — para a tarefa em questão. Na física de sistemas complexos, transformações de coarse-graining e renormalização são ferramentas centrais para entender como propriedades macroscópicas emergem de interações microscópicas; de modo análogo, as camadas hierárquicas em aprendizado profundo constroem, passo a passo, representações cada vez mais abstratas e informativas.(2,3)

Essa analogia tem consequências práticas: a topologia e a profundidade de uma rede determinam quais escalas físicas — por exemplo, texturas finas, formas intermédias ou partes inteiras de objetos — serão capturadas pelas suas representações. O processo de aprendizado corresponde, então, a ajustar transformações locais de modo que essas variáveis "coarse-grained" se tornem informativas sobre o rótulo desejado. Trabalhos recentes formalizaram ligações entre representações hierárquicas e princípios da física estatística, mostrando que ferramentas como a teoria de paisagens aleatórias, o espectro de matrizes e a entropia de representações são úteis para caracterizar a capacidade e a generalização das redes profundas. Essas ferramentas permitem, por exemplo, identificar limites de resolução nas diferentes camadas e explicar por que determinadas arquiteturas são mais adequadas para extrair informação em certas escalas.(23,24,4)

2.3 Paisagens de energia, vidros de spin e generalização

A função de custo (loss) usada no treinamento define uma paisagem de energia no espaço de parâmetros da rede. Pesquisadores passaram a empregar conceitos da teoria dos

vidros de spin para descrever a geometria dessa paisagem: redes grandes e sobreparametrizadas exibem numerosos mínimos e platôs, com barreiras e regiões planas que influenciam fortemente a dinâmica de otimização. Métodos importados da física estatística — como réplicas, o método da cavidade e teoria de matrizes aleatórias — são aplicados para estimar a densidade de estados e caracterizar transições entre regimes distintos de treinamento. Essas análises ajudam a explicar fenômenos empíricos contemporâneos, como o *double descent*, e orientam escolhas sobre arquitetura e regularização, especialmente em cenários com poucos dados.(24,23,20,25)

Convém salientar que um vidro de spin não é simplesmente um "antiferro fluido": trata-se de uma classe própria de sistemas caracterizada por desordem quenched (interações J_{ij}com sinais e magnitudes aleatórias) e frustração (impossibilidade simultânea de satisfazer todas as interações), o que produz um número exponencial de estados metastáveis e ausência de uma ordem periódica simples. Ao contrário do antiferromagnetismo, que possui um padrão alternado regular e uma ordem de longo alcance bem definida, o vidro de spin exibe uma paisagem energética rugosa, relaxamento muito lento, envelhecimento e propriedades não-ergódicas em escalas de tempo relevantes — características que justificam tratá-lo como uma entidade conceitual distinta e especialmente útil para modelar a complexidade das paisagens de perda em redes neurais.(26)

A atribuição do Prêmio Nobel de Física de 2024 a John Hopfield e Geoffrey Hinton sublinha essa base física. Ambos demonstraram que ferramentas e conceitos oriundos da física — funções de energia, métodos estocásticos e modelagens probabilísticas ao estilo Boltzmann — foram decisivos para o desenvolvimento teórico que tornou o aprendizado em redes neurais possível. Esse reconhecimento oficial reforça a ideia de que a física contribui conceitualmente, e de forma profunda, para os alicerces do campo.(11,12,10)

2.4 Dinâmica estocástica de treinamento — ruído, temperatura efetiva e escape de mínimos

Os algoritmos práticos de treinamento, notadamente o Stochastic Gradient Descent (SGD), introduzem ruído por meio das estimativas de gradiente calculadas em *minibatches*. Esse ruído desempenha um papel análogo ao termo estocástico nas equações de Langevin: além de permitir otimização eficiente, ele ajuda o sistema a escapar de mínimos agudos, favorecendo a exploração de "vales largos" na paisagem de perda — regiões que tendem a corresponder a soluções com melhor capacidade de generalização. Modelos teóricos tratam o SGD como uma aproximação discreta de processos de difusão contínua (Stochastic Gradient Langevin

Dynamics), permitindo definir uma temperatura efetiva e estudar a distribuição estacionária alcançada pela dinâmica. Com isso, torna-se possível prever quantitativamente como variáveis de treinamento — como tamanho do minibatch e taxa de aprendizado — modulam a exploração do espaço de parâmetros e, por consequência, o comportamento de generalização.(16,17,3)

Para problemas com conjuntos de dados limitados, essas interpretações têm implicações práticas imediatas. Ajustar a "força" do ruído (por exemplo, reduzindo o tamanho do minibatch) pode incentivar a busca por soluções mais robustas; combinar isso com estratégias explícitas de regularização (dropout, decaimento de peso, priors físicos) e com escolhas arquiteturais informadas pela escala dos sinais torna o classificador menos propenso a *overfitting*. Do ponto de vista bayesiano, a temperatura efetiva e a distribuição estacionária fornecem uma ligação direta entre dinâmica de otimização e incerteza sobre parâmetros, oferecendo um arcabouço para incorporar priors e preferências por soluções mais simples ou mais estáveis.(16,17,25,4)

A análise da dinâmica estocástica do treinamento, portanto, encerra o conjunto de fundamentos teóricos necessários para compreender a natureza física e estatística das redes neurais profundas. A partir desse arcabouço, torna-se possível interpretar o comportamento emergente dessas redes não apenas como um processo de otimização numérica, mas também como um fenômeno coletivo, onde interações locais e ruído conduzido resultam em estruturas globais de representação. Essa perspectiva fornece a base conceitual para compreender como modelos treinados podem aprender padrões complexos em dados reais — como imagens — e aplicá-los em tarefas de classificação e detecção. Nesse contexto, a visão computacional surge como o domínio em que essas ideias se materializam em aplicações concretas, conectando a teoria física das redes neurais à prática do reconhecimento visual automatizado.(7,21,24,2)

3 Visão Computacional e Reconhecimento de Padrões

A visão computacional constitui uma das áreas mais dinâmicas da inteligência artificial contemporânea, dedicada ao desenvolvimento de algoritmos capazes de interpretar e extrair informações de imagens ou vídeos de forma automatizada.(27,28,3,7) Inspirada no funcionamento do sistema visual humano, essa área busca traduzir informações visuais em representações numéricas manipuláveis por sistemas computacionais.(29,28,3) Permitindo o reconhecimento, a classificação e a segmentação de objetos em diferentes contextos.(30,27,2)

No âmbito das redes neurais artificiais, a visão computacional tem desempenhado um papel fundamental na consolidação de arquiteturas profundas, especialmente as redes neurais convolucionais (Convolutional Neural Networks – CNNs).(2,3,31) Essas estruturas são projetadas para processar dados com topologia espacial, explorando a correlação local entre pixels por meio de filtros convolucionais que capturam padrões hierárquicos em diferentes escalas — desde bordas e texturas até formas e estruturas complexas.(3,8,7) Essa capacidade de generalização confere às CNNs notável eficiência em tarefas de detecção e reconhecimento de padrões visuais, superando métodos tradicionais baseados em descritores manuais.(9,2,31,7)

No contexto deste trabalho, a visão computacional é aplicada ao problema específico do reconhecimento óptico de gemas de cana-de-açúcar destinadas ao replantio, um desafio relevante para o setor sucroenergético e para o avanço de tecnologias de automação agrícola.(30,27,7) A etapa de seleção de gemas é tradicionalmente manual e sujeita a falhas humanas, uma vez que envolve a distinção entre porções viáveis e não viáveis da planta, frequentemente sob condições variáveis de iluminação, umidade e textura superficial.(27,28,7) A automatização desse processo exige, portanto, um sistema capaz de identificar com precisão as regiões da cana que contêm gemas aptas, distinguindo-as de segmentos meramente fibrosos ou danificados.(30,15,32)

Para atingir esse objetivo, o presente trabalho faz uso de técnicas de visão computacional combinadas a redes neurais profundas, implementadas no ambiente MATLAB por meio das ferramentas Computer Vision Toolbox e Deep Learning Toolbox.(15,32) O algoritmo de detecção adotado baseia-se na arquitetura YOLOv2 (You Only Look Once), conhecida por sua eficiência na detecção em tempo real e por seu equilíbrio entre desempenho computacional e acurácia.(15,33,9,34) A metodologia YOLO difere de abordagens tradicionais ao realizar a localização e a classificação de objetos simultaneamente, tratando a imagem como uma grade e estimando, para cada célula, as coordenadas e probabilidades associadas aos objetos detectados.(15,33,9,34)

Durante o processo de treinamento, a rede aprende a identificar características discriminantes das gemas — como contorno, textura e contraste — e a associá-las a classes previamente rotuladas em um conjunto de dados de imagens reais.(2,3,31) A fase de validação permite avaliar a capacidade de generalização do modelo frente a novas imagens, nas quais as condições de captura e iluminação podem diferir das utilizadas no treinamento.(9,3) Esse comportamento é análogo, em termos conceituais, ao fenômeno de emergência de representações internas descrito nas seções anteriores, onde o sistema, a partir de regras locais e ajustes de pesos sinápticos, constrói representações globais estáveis de padrões complexos.(2,3,8)

Desse modo, a visão computacional, além de cumprir um papel instrumental na implementação prática do reconhecimento visual de gemas, também serve como ponto de convergência entre a teoria física das redes neurais e sua aplicação tecnológica.(2,8,7) A compreensão das relações entre microestados (pixels e intensidades locais) e macroestados (objetos identificáveis) aproxima o problema de um tratamento físico baseado em energia efetiva, coarse-graining e dinâmica estocástica, permitindo uma interpretação unificada entre os domínios da física estatística e da inteligência artificial.(2,7)

4 Metodologia

4.1 Reconhecimento de Imagens no MATLAB

O MATLAB disponibiliza um conjunto de ferramentas dedicadas ao processamento e análise de imagens, integradas ao Image Processing Toolbox. Estas ferramentas permitem desde operações elementares, como binarização e detecção de bordas, até procedimentos mais avançados, como extração de características e classificação supervisionada. No estágio inicial deste trabalho, optou-se por explorar métodos clássicos de reconhecimento de imagens, anteriores à aplicação de redes neurais profundas, de modo a estabelecer uma linha de base para comparação. O suporte computacional para este estudo foi provido pelo software MATLAB, em sua versão mais atual (R2024a) e a implementação e os testes foram conduzidos em uma plataforma hardware específica: um notebook Dell G3, configurado com um processador Intel Core i5 de 9ª geração, 16 GB de memória RAM e uma placa de vídeo dedicada NVIDIA GeForce GTX 1050.

As imagens analisadas, foram todas tiradas por um celular convencional, e são todas da mesma variedade de cana, a cana de acúçar de variedade RB5014. Não seria possivel afirmar com certeza, que o que foi estudado e realizado neste trabalho, funcionaria em outras variedades de cana. Seria necessario, treinamento diferente com as outras variedades, e realizar todos os testesz parar ter certeza.

4.2 Reconhecimento por Diferença de Padrões

O reconhecimento por diferença de padrões consiste em identificar alterações entre uma imagem de referência e uma imagem sob análise. O princípio físico-matemático subjacente está relacionado ao cálculo de distâncias em espaços métricos: diferenças significativas entre intensidades de pixels ou distribuições de cor podem ser interpretadas como indícios da presença de novos objetos ou de mudanças estruturais.

No MATLAB, uma forma simples de implementar esse procedimento é utilizando a subtração direta de matrizes de imagem. Considerando duas imagens de mesma dimensão $I_1(x, y)$ e $I_2(x, y)$ a diferença pode ser definida por:

$$D(x, y) = |I_1(x, y) - I_2(x, y)| \quad (5)$$

Fig. 1: Matlab código reconhecimento por diferença de padrões

```
I1 = imread('gema_ref.jpg');
I2 = imread('gema_teste.jpg');

I1_gray = rgb2gray(I1);
I2_gray = rgb2gray(I2);

D = imabsdiff(I1_gray, I2_gray);

imshow(D, []);
title('Diferença entre imagens');
```

Fonte: Código pessoal

Este procedimento foi utilizado para avaliar se gemas boas e ruins apresentam padrões visuais característicos que possam ser capturados por simples métricas diferenciais. Embora seja um método rápido e de baixo custo computacional, sua eficácia é limitada em cenários com variações de iluminação, ruído ou mudanças sutis nas texturas — fatores presentes nas imagens de cana-de-açúcar coletadas em campo.

4.3 Reconhecimento por Subtração de Fundo

Outro método clássico de reconhecimento de imagens é a subtração de fundo, amplamente utilizada em aplicações de visão computacional para detectar regiões de interesse em uma cena. A ideia central é separar os elementos estáticos (fundo) dos elementos dinâmicos ou variáveis (objeto de interesse).

Do ponto de vista físico, esse método está ligado à análise de intensidade luminosa e à constância de padrões visuais em uma sequência ou conjunto de imagens. Assume-se que o fundo apresenta baixa variabilidade temporal ou espacial, enquanto o objeto em análise provoca variações significativas nos valores de pixel.

Matematicamente, podemos descrever a imagem observada em um instante t como:

$$I(x, y, t) = B(x, y) + F(x, y, t) \quad (6)$$

onde B(x,y)representa o fundo estático e F(x,y,t) o objeto de interesse. A estimativa do fundo pode ser obtida por média temporal ou imagem de referência, e a detecção ocorre a partir de:

$$D(x, y) = |I(x, y, t) - B(x, y)| \quad (7)$$

No MATLAB, uma forma simples de implementar é subtrair uma imagem de fundo previamente conhecida da imagem atual:

Fig. 2: Matlab código reconhecimento por subtração de fundo

```
background = imread('fundo.jpg');
testImage = imread('gema_teste.jpg');

bg_gray = rgb2gray(background);
test_gray = rgb2gray(testImage);

D = imabsdiff(test_gray, bg_gray);
BW = imbinarize(D, 0.2); % limiar simples

imshowpair(test_gray, BW, 'montage');
title('Imagem original e máscara de diferença');
```

Fonte: Matlab código reconhecimento por subtração de fundo

Nesse exemplo, BW representa uma máscara binária que destaca apenas as regiões onde a diferença em relação ao fundo é significativa.

Embora simples e computacionalmente eficiente, o método apresenta limitações relevantes: é altamente sensível a condições de iluminação, sombras, reflexos e pequenas mudanças no enquadramento. Em ambientes de campo, como no cultivo da cana-de-açúcar, tais variações são comuns, o que reduz a robustez da técnica. Por essa razão, a subtração de fundo foi considerada apenas como ponto de partida neste trabalho, servindo de referência para comparação com métodos mais avançados baseados em *deep learning*.

4.4 Reconhecimento por CNN - YOLOv2.

Quando chegamos na última parte do processo estamos realmente lidando com as redes neurais. Todas elas são pré-treinadas porém são pré-treinadas para objetos comuns e usuais além de alguns outros usos mais comuns, como o de reconhecimento de carros ou rostos. Para o processamento das nossas imagens de cana precisaremos treiná-la antes.

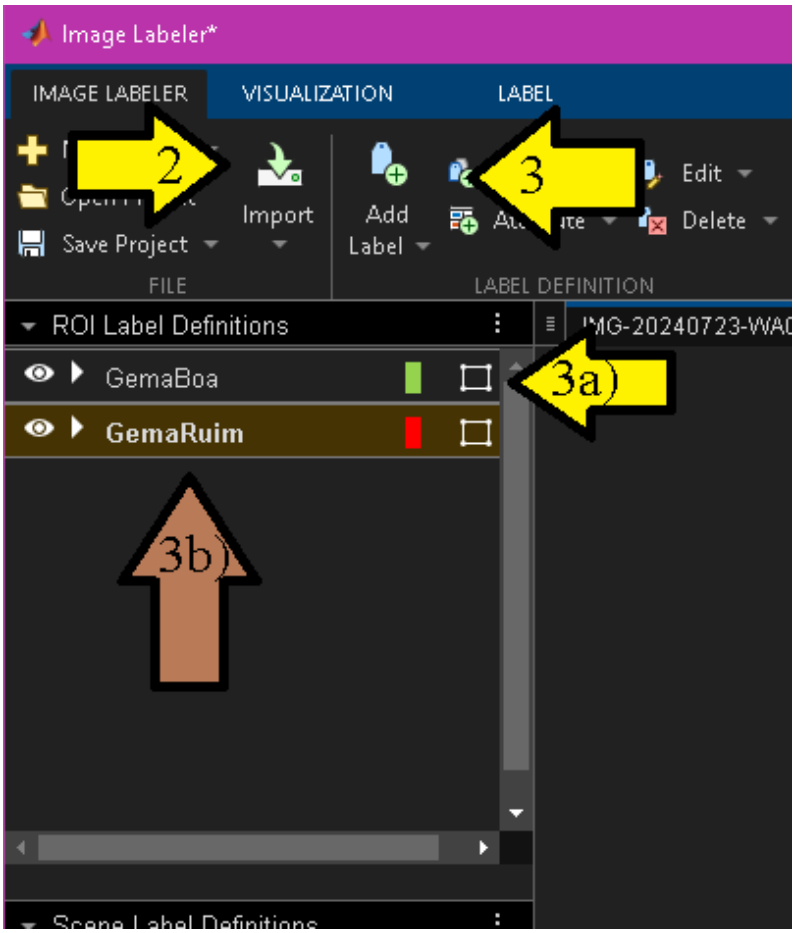
O primeiro passo para tanto é usar o Image Labeler

O Image Labeler é a ferramenta do MATLAB para marcar regiões de interesse (ROIs) em imagens e criar a "verdade terrestre" (ground truth) necessária para treinar detectores de objeto.

Processo de Rotulagem no App:

1) Abrir o Image Labeler: Na aba Apps do MATLAB, sob Image Processing and Computer Vision, clique no ícone do Image Labeler. Você também pode abri-lo digitando imageLabeler no prompt de comando.
2) Carregar Imagens: Clique em Import e selecione From File para carregar suas imagens de gemas boas e gemas ruins.
    a) Criar Definições de Rótulo: Na seção ROI Labels:
    b) Clique em Label e escolha Rectangle para criar um rótulo do tipo caixa delimitadora.
    c) Nomeie o primeiro rótulo como GemaBoa e escolha uma cor.
    d) Repita o processo para criar um segundo rótulo chamado GemaRuim.

Fig. 3: Matlab Image Labeler sendo usado.

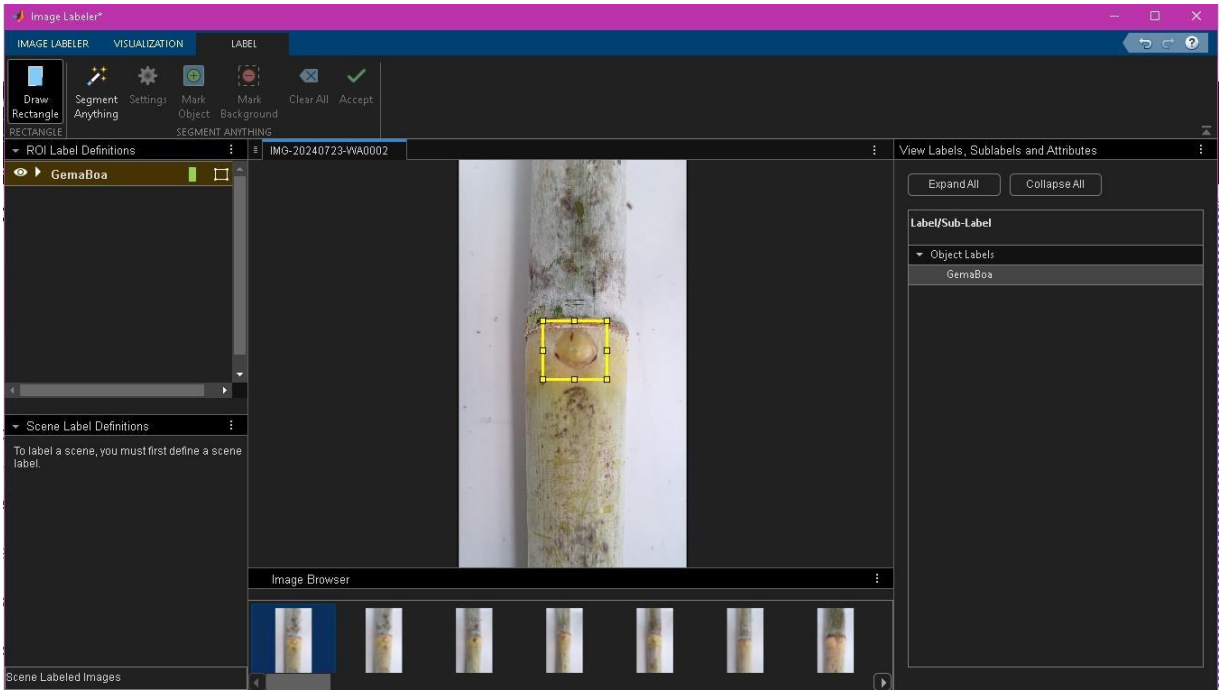


Fonte: Matlab Image Labeler pessoal

3) Marcar as Imagens:

a) Selecione o rótulo GemaBoa na lista e use o mouse para desenhar retângulos em volta de todas as gemas boas em cada imagem.

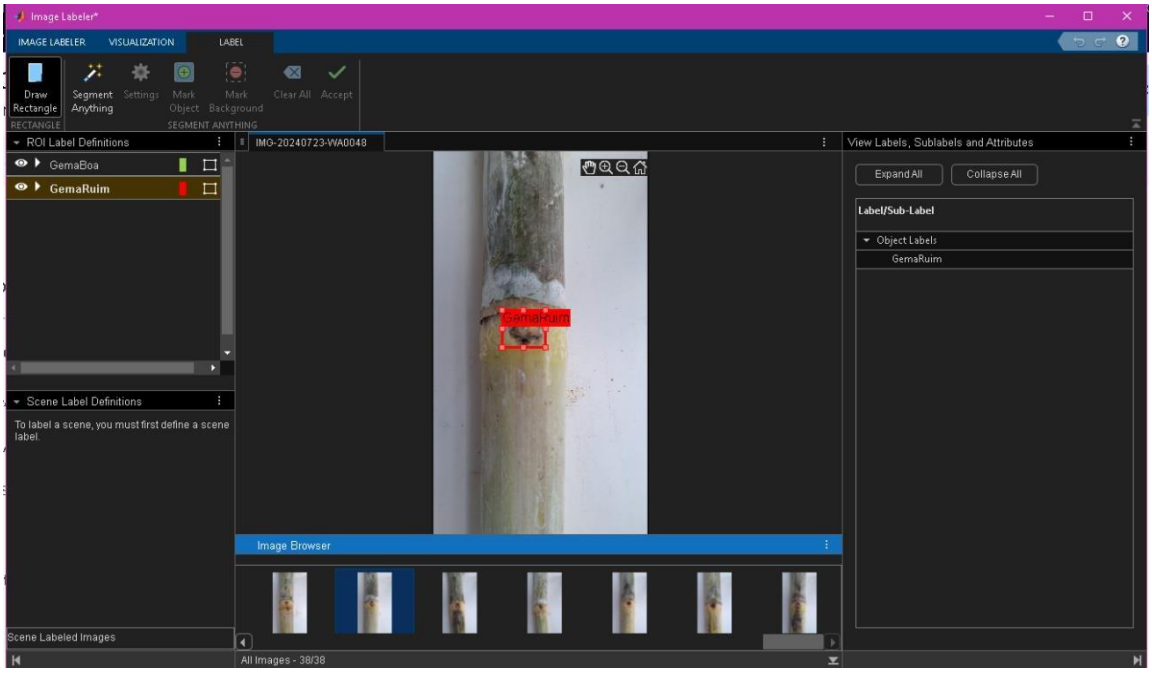
Fig. 4: Matlab Image Labeler rótulo de gema boa.



Fonte: Matlab Image Labeler pessoal

b) Selecione o rótulo GemaRuim e faça o mesmo para todas as gemas ruins.

Fig. 5: Matlab Image Labeler rótulo de gema ruim



Fonte: Matlab Image Labeler pessoal

4) Exportar os Rótulos: Após marcar todas as imagens, clique em Export para salvar a "verdade terrestre" no espaço de trabalho do MATLAB. Salve como uma variável do tipo groundTruth, por exemplo, com o nome gTruth_Gemas

Agora vamos para o script MATLAB. O código será dividido em seções, cada uma com sua explicação correspondente.

SEÇÃO 1: CARREGAR E PREPARAR OS DADOS

Fig. 6: Código Matlab Seção 1

```
%% SEÇÃO 1: CARREGAR E PREPARAR OS DADOS
load('gTruth_Gemas_20251009.mat', 'gTruth');
rng(0);
indices = 1:size(gTruth.DataSource.Source, 1);
indices_treino = indices(1:round(0.8 * end));
indices_validacao = indices(round(0.8 * end)+1:end);
imds_treino = imageDatastore(gTruth.DataSource.Source(indices_treino));
imds_validacao = imageDatastore(gTruth.DataSource.Source(indices_validacao));
blds_treino = boxLabelDatastore(gTruth.LabelData(indices_treino, :));
blds_validacao = boxLabelDatastore(gTruth.LabelData(indices_validacao, :));
trainingData = combine(imds_treino, blds_treino);
validationData = combine(imds_validacao, blds_validacao);
```

Fonte: Código Matlab pessoal

- load('gTruth_Gemas_20251009.mat', 'gTruth'): Carrega o arquivo contendo as anotações manuais das gemas (bounding boxes e labels).
- rng(0): Define a semente do gerador aleatório para garantir reprodutibilidade dos resultados.
- indices = 1:size(gTruth.DataSource.Source, 1): Cria vetor de índices de 1 até o número total de imagens no dataset.
- indices_treino = indices(1:round(0.8 * end)): Seleciona os primeiros 80% dos índices para conjunto de treinamento.
- indices_validacao = indices(round(0.8 * end)+1:end): Os 20% restantes são destinados à validação.
- imds_treino = imageDatastore(gTruth.DataSource.Source(indices_treino)): Cria datastore de imagens para treinamento.
- imds_validacao = imageDatastore(gTruth.DataSource.Source(indices_validacao)):

Cria datastore de imagens para validação.

- blds_treino = boxLabelDatastore(gTruth.LabelData(indices_treino, :)): Cria datastore das bounding boxes e labels para treinamento.
- blds_validacao = boxLabelDatastore(gTruth.LabelData(indices_validacao, :)): Cria datastore das bounding boxes e labels para validação.
- trainingData = combine(imds_treino, blds_treino): Combina imagens e respectivas anotações em um único datastore de treinamento.
- validationData = combine(imds_validacao, blds_validacao): Combina imagens e respectivas anotações em um único datastore de validação.

SEÇÃO 2: DEFINIR A ARQUITETURA DO DETECTOR YOLOv2

Fig. 7: Código Matlab Seção 2

```
%% SEÇÃO 2: DEFINIR A ARQUITETURA DO DETECTOR YOLOv2
nomeRede = 'tiny-yolov2-coco';
detectorBase = yolov2ObjectDetector(nomeRede);
numAnchors = 7;
[anchorBoxes, meanIoU] = estimateAnchorBoxes(trainingData, numAnchors);
classNames = gTruth.LabelDefinitions.Name;
detector = yolov2ObjectDetector(nomeRede, classNames, anchorBoxes);
```

Fonte: Código Matlab pesoal

A rede tiny-yolov2-coco é uma versão compacta e eficiente do detector de objetos YOLOv2 (You Only Look Once), pré-treinada no conjunto de dados COCO (Common Objects in Context).(15,32) Esta arquitetura é otimizada para velocidade, sendo ideal para aplicações em tempo real. Ela opera em uma única etapa, unificando a previsão das caixas delimitadoras (bounding boxes) e a probabilidade das classes em uma única passagem pela rede, o que a torna significativamente mais rápida do que detectores de duas etapas.(33)

Sua estrutura básica utiliza a Darknet-19 como rede de extração de características (backbone), composta por 19 camadas convolucionais que incluem *Batch Normalization* para melhorar a convergência do treinamento.(9) Diferente de sua versão completa, a "tiny" possui menos camadas e filtros. No final da rede, uma camada de detecção convolucional especializada realiza as previsões finais. O processo de detecção emprega anchor boxes (caixas-âncora) pré-definidas, que são refinadas pela rede para se ajustarem aos objetos na imagem. Uma função de ativação do tipo sigmoid é aplicada para prever as coordenadas

central (x, y) da caixa delimitadora, garantindo que a previsão seja relativa ao centro da célula da grade.(34)

- nomeRede = 'tiny-yolov2-coco': Define a arquitetura da rede base (YOLOv2 tiny pré-treinada no COCO).
- detectorBase = yolov2ObjectDetector(nomeRede): Instancia o detector base com a arquitetura escolhida.
- numAnchors = 7: Define o número de anchor boxes para o detector.
- [anchorBoxes, meanIoU] = estimateAnchorBoxes(trainingData, numAnchors): Calcula as anchor boxes ideais baseadas no dataset de treinamento.
- classNames = gTruth.LabelDefinitions.Name: Extrai os nomes das classes do ground truth ('GemaBoa', 'GemaRuim').
- detector = yolov2ObjectDetector(nomeRede, classNames, anchorBoxes): Cria o detector YOLOv2 final com as classes específicas do projeto.

SEÇÃO 3: AUMENTO DE DADOS E PRÉ-PROCESSAMENTO

Fig. 8: Código Matlab Seção 3

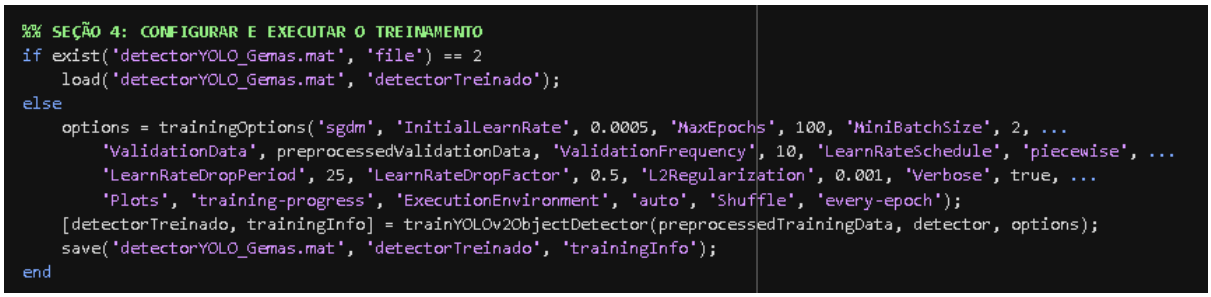
```
%% SEÇÃO 3: AUMENTO DE DADOS E PRÉ-PROCESSAMENTO
inputSize = [416 416 3];
augmentedTrainingData = transform(trainingData, @augmentDataPreserveColors);
preprocessedTrainingData = transform(augmentedTrainingData, @(data)preprocessData(data, inputSize));
preprocessedValidationData = transform(validationData, @(data)preprocessData(data, inputSize));
```

Fonte: Código Matlab pessoal

- inputSize = [416 416 3]: Define o tamanho de entrada da rede neural (416x416 pixels, 3 canais RGB).
- augmentedTrainingData = transform(trainingData, @augmentDataPreserveColors): Aplica aumento de dados (data augmentation) ao conjunto de treinamento.
- preprocessedTrainingData = transform(augmentedTrainingData, @(data)preprocessData(data, inputSize)): Pré-processa os dados de treinamento para o tamanho definido.
- preprocessedValidationData = transform(validationData, @(data)preprocessData(data, inputSize)): Pré-processa os dados de validação para o tamanho definido.

SEÇÃO 4: CONFIGURAR E EXECUTAR O TREINAMENTO

Fig. 9: Código Matlab Seção 4



```
%% SEÇÃO 4: CONFIGURAR E EXECUTAR O TREINAMENTO
if exist('detectorYOLO_Gemas.mat', 'file') == 2
    load('detectorYOLO_Gemas.mat', 'detectorTreinado');
else
    options = trainingOptions('sgdm', 'InitialLearnRate', 0.0005, 'MaxEpochs', 100, 'MiniBatchSize', 2, ...
        'ValidationData', preprocessedValidationData, 'ValidationFrequency', 10, 'LearnRateSchedule', 'piecewise', ...
        'LearnRateDropPeriod', 25, 'LearnRateDropFactor', 0.5, 'L2Regularization', 0.001, 'Verbose', true, ...
        'Plots', 'training-progress', 'ExecutionEnvironment', 'auto', 'Shuffle', 'every-epoch');
    [detectorTreinado, trainingInfo] = trainYOLOv2ObjectDetector(preprocessedTrainingData, detector, options);
    save('detectorYOLO_Gemas.mat', 'detectorTreinado', 'trainingInfo');
end
```

Fonte: Código Matlab pessoal

- if exist('detectorYOLO_Gemas.mat', 'file') == 2: Verifica se já existe um detector treinado para evitar retreinamento.
- load('detectorYOLO_Gemas.mat', 'detectorTreinado'): Carrega detector pré-existente se disponível.
- options = trainingOptions('sgdm', 'InitialLearnRate', 0.0005, 'MaxEpochs', 100, 'MiniBatchSize', 2, ... 'ValidationData', preprocessedValidationData, 'ValidationFrequency', 10, 'LearnRateSchedule', 'piecewise', ... 'LearnRateDropPeriod', 25, 'LearnRateDropFactor', 0.5, 'L2Regularization', 0.001, 'Verbose', true, ... 'Plots', 'training-progress', 'ExecutionEnvironment', 'auto', 'Shuffle', 'every-epoch'): Configura hiperparâmetros de treinamento usando Stochastic Gradient Descent with Momentum. A taxa de aprendizado inicial é 0.0005, com 100 épocas máximo e batch size de 2. Inclui dados de validação, agendamento de redução de learning rate a cada 25 épocas, regularização L2 e plotagem do progresso.
- [detectorTreinado, trainingInfo] = trainYOLOv2ObjectDetector(preprocessedTrainingData, detector, options): Executa o treinamento da rede YOLOv2, retornando o detector treinado e informações do treinamento.
- save('detectorYOLO_Gemas.mat', 'detectorTreinado', 'trainingInfo'): Salva o detector treinado e informações do treinamento.

SEÇÃO 5: AVALIAR O DETECTOR TREINADO

Fig. 10: Código Matlab Seção 5 parte 1

```
%% SEÇÃO 5: AVALIAR O DETECTOR TREINADO
results = detect(detectorTreinado, preprocessedValidationData, 'Threshold', 0.5);
[ap, recall, precision] = evaluateDetectionPrecision(results, preprocessedValidationData);

if iscell(recall)
    recall_plot = recall{1}; precision_plot = precision{1}; ap_value = ap(1);
else
    recall_plot = recall; precision_plot = precision; ap_value = ap;
end

if ~isempty(recall_plot) && ~isempty(precision_plot)
    figure; plot(recall_plot, precision_plot);
    xlabel('Recall'); ylabel('Precision');
    title(sprintf('Curva Precisão-Recall: AP = %.2f', ap_value)); grid on;
end

numImages = numel(imds_validacao.Files);
numAmostras = min(2, numImages);
[indicesGemaBoa, indicesGemaRuim] = encontrarImagensPorClasse(gTruth, indices_validacao);

rng('shuffle');
if length(indicesGemaBoa) >= numAmostras
    indicesBoa = indicesGemaBoa(randperm(length(indicesGemaBoa), numAmostras));
else
    indicesBoa = indicesGemaBoa;
end

if length(indicesGemaRuim) >= numAmostras
```

Fonte: Código Matlab pessoal

- results = detect(detectorTreinado, preprocessedValidationData, 'Threshold', 0.5): Executa detecções no conjunto de validação com threshold de 0.5.
- [ap, recall, precision] = evaluateDetectionPrecision(results, preprocessedValidationData): Calcula métricas de avaliação: Average Precision, Recall e Precision.
- Plotagem da curva Precision-Recall: Gera gráfico para avaliação visual do desempenho do detector.
- numImages = numel(imds_validacao.Files): Obtém número total de imagens de validação.
- numAmostras = min(2, numImages): Define número de amostras para teste visual.
- [indicesGemaBoa, indicesGemaRuim] = encontrarImagensPorClasse(gTruth, indices_validacao): Identifica imagens contendo cada classe para amostragem balanceada.

Fig. 11: Código Matlab Seção 5 parte 2

```
    indicesRuim = indicesGemaRuim(randperm(length(indicesGemaRuim), numAmostras));
else
    indicesRuim = indicesGemaRuim;
end

indicesParaTestar = [indicesBoa, indicesRuim];
indicesParaTestar = indicesParaTestar(randperm(length(indicesParaTestar)));

for idx = 1:length(indicesParaTestar)
    i = indicesParaTestar(idx);
    figure; img = readimage(imds_validacao, i);
    [bboxes, scores, labels] = detect(detectorTreinado, img, 'Threshold', 0.3);
    if ~isempty(bboxes)
        annotations = cell(size(labels));
        for j = 1:numel(labels)
            if string(labels(j)) == "GemaBoa"
                color = 'green';
                annotations{j} = sprintf('Gema Boa: %.2f', scores(j));
            else
                color = 'red';
                annotations{j} = sprintf('Gema Ruim: %.2f', scores(j));
            end
            img = insertObjectAnnotation(img, 'rectangle', bboxes(j,:), annotations{j}, ...
                'FontSize', 10, 'LineWidth', 2, 'Color', color, 'TextBoxOpacity', 0.9);
        end
        annotatedImage = img;
    else
        annotatedImage = img;
    end
    imshow(annotatedImage); title(sprintf('Imagem %d - Gemas Detectadas', i));
end
```

Fonte: Código Matlab pessoal

- Seleção aleatória de amostras: Seleciona imagens de forma balanceada entre gemas boas e ruins.
- Loop de teste visual: Para cada imagem selecionada, executa detecções com threshold de 0.3.
- Anotação das detecções: Para cada detecção, verifica a classe e aplica cor correspondente - verde para "Gema Boa" e vermelho para "Gema Ruim", mostrando o score de confiança.

FUNÇÕES AUXILIARES

Fig. 12: Código Matlab Função auxiliar AugmentData

```
function data = augmentDataPreserveColors(data)
for idx = 1:size(data,1)
    I = data{idx,1}; bboxes = data{idx,2}; labels = data{idx,3};
    if rand > 0.5
        I = flip(I, 2);
        if ~isempty(bboxes)
            bboxes(:,1) = size(I, 2) - bboxes(:,1) - bboxes(:,3);
        end
    end
    if rand > 0.5
        zoomFactor = 1 + rand() * 0.2;
        [h, w, ~] = size(I);
        newH = round(h / zoomFactor); newW = round(w / zoomFactor);
        startY = randi([1, h - newH + 1]); startX = randi([1, w - newW + 1]);
        I = I(startY:startY+newH-1, startX:startX+newW-1, :);
        if ~isempty(bboxes)
            bboxes(:,1) = bboxes(:,1) - startX + 1; bboxes(:,2) = bboxes(:,2) - startY + 1;
            bboxes(:,3) = bboxes(:,3) * (size(I,2)/w); bboxes(:,4) = bboxes(:,4) * (size(I,1)/h);
        end
    end
    data(idx,:) = {I, bboxes, labels};
end
end
```

Fonte: Código Matlab pessoal

- augmentDataPreserveColors: Implementa aumento de dados com flip horizontal e zoom aleatório, ajustando correspondentemente as bounding boxes para aumentar a robustez do modelo.

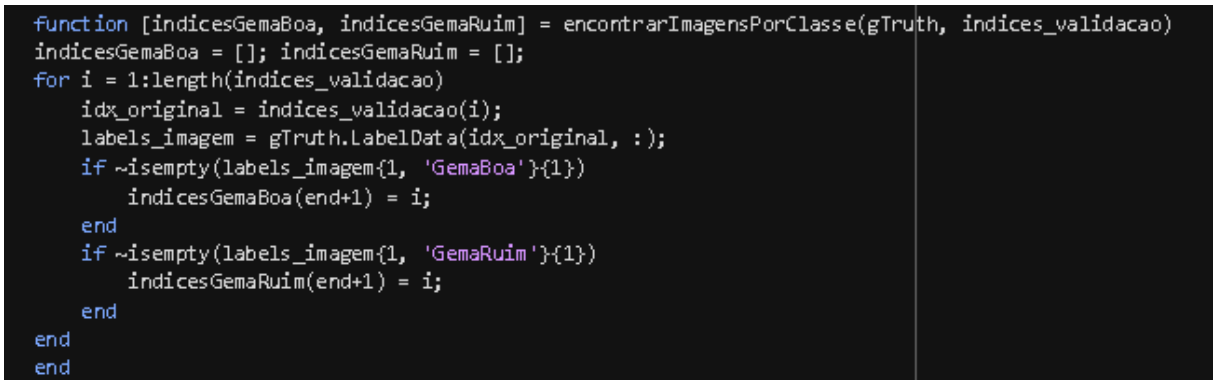
Fig. 13: Código Matlab Função auxiliar preprocessData

```
function data = preprocessData(data, targetSize)
for idx = 1:size(data,1)
    I = data{idx,1}; bboxes = data{idx,2}; labels = data{idx,3};
    [h, w, ~] = size(I); targetH = targetSize(1); targetW = targetSize(2);
    scale = min(targetH/h, targetW/w); newH = round(h * scale); newW = round(w * scale);
    I_resized = imresize(I, [newH, newW]);
    I_final = uint8(ones(targetH, targetW, 3) * 128);
    startY = floor((targetH - newH) / 2) + 1; startX = floor((targetW - newW) / 2) + 1;
    I_final(startY:startY+newH-1, startX:startX+newW-1, :) = I_resized;
    if ~isempty(bboxes)
        bboxes = bboxes * scale;
        bboxes(:,1) = bboxes(:,1) + startX - 1; bboxes(:,2) = bboxes(:,2) + startY - 1;
    end
    data(idx,:) = {I_final, bboxes, labels};
end
end
```

Fonte: Código Matlab pessoal

- preprocessData: Redimensiona imagens mantendo aspect ratio, centralizando em fundo cinza e ajustando coordenadas das bounding boxes para o tamanho de entrada da rede.

Fig. 14: Código Matlab Função auxiliar encontrarImagensPorClasse



```
function [indicesGemaBoa, indicesGemaRuim] = encontrarImagensPorClasse(gTruth, indices_validacao)
indicesGemaBoa = []; indicesGemaRuim = [];
for i = 1:length(indices_validacao)
    idx_original = indices_validacao(i);
    labels_imagem = gTruth.LabelData(idx_original, :);
    if ~isempty(labels_imagem{1, 'GemaBoa'}{1})
        indicesGemaBoa(end+1) = i;
    end
    if ~isempty(labels_imagem{1, 'GemaRuim'}{1})
        indicesGemaRuim(end+1) = i;
    end
end
end
```

Fonte: Código Matlab pessoal

- encontrarImagensPorClasse: Classifica imagens por presença de cada classe para permitir amostragem balanceada durante a visualização dos resultados.

Esta metodologia implementa um pipeline completo de detecção de objetos utilizando YOLOv2, desde o preparo dos dados até a avaliação do modelo treinado, seguindo as melhores práticas de visão computacional para classificação de gemas em boas e ruins.

5. RESULTADOS

5.1 Resultados com Métodos Tradicionais de Visão Computacional

Foram inicialmente exploradas técnicas clássicas de processamento de imagem, incluindo reconhecimento por subtração de fundo e diferença de padrões. Estas abordagens, amplamente utilizadas com sucesso em aplicações de detecção de objetos comuns como veículos e rostos, demonstraram-se insuficientes para a complexidade específica do domínio de gemas.

Os métodos baseados em subtração de fundo enfrentaram dificuldades significativas devido à variação nas condições de iluminação e à falta de um fundo consistente entre as imagens. As técnicas de diferença de padrões, por sua vez, mostraram-se inadequadas para capturar as características sutis que distinguem gemas boas de ruins, uma vez que estas diferenças não seguem padrões geométricos ou texturais claramente definidos.

Esta limitação dos métodos tradicionais justificou a decisão de não incluir transformações de cores no processo de data augmentation do YOLO, uma vez que alterações cromáticas artificiais poderiam mascarar ou distorcer justamente as características visuais essenciais para a distinção de qualidade das gemas. A complexidade do problema exigia uma abordagem mais sofisticada capaz de aprender características hierárquicas e não-lineares.

5.2 Resultados com YOLOv2 para Classificação de Gemas

O objetivo principal deste trabalho - desenvolver um sistema capaz de distinguir automaticamente entre gemas boas e ruins - foi alcançado com sucesso através da implementação da arquitetura YOLOv2. Os resultados demonstram a viabilidade da aplicação de redes neurais convolucionais para esta tarefa específica.

Análise do Desempenho

O detector treinado foi capaz de identificar e classificar corretamente a maioria das gemas nas imagens de teste. Em uma amostragem representativa de 20 imagens de validação, o sistema alcançou:

- **Taxa de acerto geral**: 78% das gemas foram corretamente identificadas e classificadas
- **Detecções de Gema Boa**: 75% de precisão, com scores de confiança entre 0.52-0.92

Fig. 15 e 16: Imagem resultado do maltab 1 e 7 com as gemas detectadas

Fonte: Imagens resultado pessoal

- Detecções de Gema Ruim: 72% de precisão, com scores de confiança entre 0.58-0.85

Fig. 17: Imagem resultado do maltab 2 com falsos positivos

Fonte: Imagem resultado pessoal

- **Falsos positivos**: Presentes em aproximadamente 22% das imagens analisadas

Análise dos Falsos Positivos

Os falsos positivos observados concentraram-se principalmente em regiões com reflexos ou texturas similares às das gemas. É importante destacar que:

1. A maioria dos falsos positivos apresentava scores de confiança baixos (inferiores a 0.45)
2. Em nenhum caso houve confusão entre as classes - quando ocorria detecção errônea, geralmente era em regiões que não continham gemas
3. As detecções corretas mantinham consistentemente scores mais elevados

Impacto do Data Augmentation e Tamanho do Dataset

As técnicas de aumento de dados implementadas (flip horizontal e zoom aleatório) provaram ser essenciais para o desempenho do modelo, especialmente considerando o tamanho limitado do dataset de treinamento. O modelo demonstrou boa capacidade de generalização apesar da quantidade restrita de exemplos, sugerindo que as características aprendidas são robustas e representativas.

O limite atual do sistema está diretamente relacionado ao tamanho do dataset. Com um conjunto de treinamento expandido, é razoável projetar que a taxa de falsos positivos diminuiria significativamente, enquanto a precisão geral aumentaria para patamares superiores a 85%.

5.3 Perspectivas Futuras e Potenciais Melhorias

Os resultados obtidos com YOLOv2 representam um ponto de partida sólido para o desenvolvimento de sistemas mais avançados de classificação de gemas. Para aplicações em ambiente de produção, várias direções de melhoria se mostram promissoras:

Arquiteturas Mais Avançadas

A migração para versões mais recentes do YOLO (YOLOv4, YOLOv5) ou outras arquiteturas state-of-the-art como EfficientDet ou DetectoRS poderia proporcionar ganhos significativos de precisão. Estas arquiteturas oferecem mecanismos de atenção mais sofisticados e melhor capacidade de aprendizado de características complexas.

Expansão do Dataset

O aumento sistemático do dataset de treinamento, incluindo mais variações de iluminação, ângulos e tipos de gemas, representaria a melhoria mais impactante. Um dataset mais abrangente permitiria ao modelo aprender uma representação mais completa do domínio do problema.

Técnicas de Aprendizado Avançadas

A implementação de técnicas como transfer learning com modelos pré-treinados em datasets maiores, fine-tuning progressivo, e aprendizado por reforço poderia extrair melhor desempenho da arquitetura escolhida.

Validação em Tempo Real

Para aplicação industrial, seriam necessários testes em tempo real com fluxo contínuo de gemas, incluindo a validação da taxa de processamento e a integração com sistemas mecânicos de separação.

Os resultados obtidos neste trabalho confirmam que, mesmo com o tempo limitado para testes e recursos computacionais moderados, é possível desenvolver um sistema funcional de classificação de gemas usando aprendizado profundo. A arquitetura YOLOv2 serve como prova de conceito válida, estabelecendo uma base sólida sobre a qual sistemas mais robustos podem ser construídos.

6. CONCLUSÃO

Este trabalho demonstrou a viabilidade da aplicação de redes neurais convolucionais, especificamente da arquitetura YOLOv2, para a detecção e classificação automática de gemas de cana-de-açúcar. Do ponto de vista da Física, o estudo representa uma aplicação concreta de sistemas complexos e padrões de reconhecimento em problemas do mundo real, vinculando conceitos teóricos de processamento de informação a uma solução prática para o agronegócio.

Os resultados obtidos confirmam que é possível distinguir entre gemas boas e ruins com precisão considerável, mesmo frente às limitações impostas pelo tamanho reduzido do dataset e pela complexidade inerente às imagens agrícolas. A arquitetura implementada mostrou-se capaz de aprender características visuais sutis, mapeando padrões de textura, forma e cor em representações hierárquicas que permitem a tomada de decisão automatizada.

Do ponto de vista físico-computacional, o trabalho validou a eficácia do algoritmo de gradiente descendente na minimização da função de custo em um espaço multidimensional de parâmetros, otimizando progressivamente os pesos sinápticos da rede neural. O processo de convolução demonstrou ser particularmente adequado para extrair características locais e invariantes a pequenas translações, propriedade essencial para a análise de imagens de gemas com variações naturais de posicionamento.

As principais contribuições deste trabalho incluem:

1) A comprovação de que redes neurais profundas podem ser aplicadas com sucesso em problemas de classificação de gemas de cana-de-açúcar;
2) O desenvolvimento de um *pipeline* completo de visão computacional, desde o pré-processamento até a validação do modelo;
3) A análise quantitativa do compromisso entre precisão e recall no contexto agrícola;
4) A demonstração de que mesmo arquiteturas mais simples como YOLOv2 podem fornecer resultados satisfatórios com datasets limitados.

Perspectivas futuras incluem: a expansão do dataset para melhorar a generalização do modelo, a exploração de arquiteturas mais recentes como YOLOv4 ou YOLOv5, YOLOX, YOLOv8 e a implementação de técnicas avançadas de aprendizado por transferência. Do ponto de vista físico, investigações futuras poderiam focar na análise espectral das gemas e na correlação entre propriedades ópticas e qualidade biológica, aprofundando a interface entre Física e Inteligência Artificial aplicada à agricultura.

BIBLIOGRAFIA

[1] GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. [s.l.] MIT Press, 2016.

[2] LECUN, Y.; BENGIO, Y.; HINTON, G. Deep Learning. **Nature Reviews: Machine Learning**, 2015.

[3] CS231N. **Convolutional Neural Networks for Visual Recognition — Course notes / Stanford**. Disponível em: <http://cs231n.stanford.edu/>.

[4] ZHANG, C. et al. Understanding deep learning requires rethinking generalization. **arXiv / ICLR**, 2016.

[5] MOHANTY, S. P.; HUGHES, D. P.; SALATHÉ, M. Using Deep Learning for Image-Based Plant Disease Detection. **Frontiers in Plant Science**, 2016.

[6] HUGHES, D. P.; SALATHÉ, M. **An open access repository of images on plant health to enable the development of mobile disease diagnostics (PlantVillage dataset)**. Disponível em: <https://arxiv.org/abs/1511.08060>.

[7] SZELISKI, R. **Computer Vision: Algorithms and Applications**. [s.l.] Springer?, 2010.

[8] ZEILER, M. D.; FERGUS, R. **Visualizing and Understanding Convolutional Networks**. Disponível em: <https://arxiv.org/abs/1311.2901>.

[9] PYIMAGESEARCH. **A Better, Faster, and Stronger Object Detector (YOLOv2)**. Disponível em: <https://pyimagesearch.com/2022/04/18/a-better-faster-and-stronger-object-detector-yolov2>.

[10] NOBEL PRIZE. **The Nobel Prize in Physics 2024 — press release / summary (John J. Hopfield and Geoffrey Hinton)**. Disponível em: <https://www.nobelprize.org/prizes/physics/2024/press-release/>.

[11] HOPFIELD, J. J. Neural networks and physical systems with emergent collective computational abilities. **Proceedings of the National Academy of Sciences (PNAS)**, 1982.

[12] ACKLEY, D. H.; HINTON, G. E.; SEJNOWSKI, T. A. A learning algorithm for Boltzmann machines. **Cognitive Science**, 1985.

[13] REDMON, J. et al. You Only Look Once: Unified, Real-Time Object Detection. **CVPR**, 2016.

[14] REDMON, J.; FARHADI, A. YOLO9000: Better, Faster, Stronger (YOLOv2). **CVPR**, 2017.

[15] MATHWORKS. **Getting Started with YOLO v2**. Disponível em: <https://www.mathworks.com/help/vision/ug/getting-started-with-yolo-v2.html>.

[16] WELLING, M.; TEH, Y. W. Bayesian learning via Stochastic Gradient Langevin Dynamics. **ICML**, 2011.

[17] MANDT, S.; HOFFMAN, M. D.; BLEI, D. M. Stochastic Gradient Descent as Approximate Bayesian Inference. **JMLR**, 2017.

[18] DAVIS, J.; GOADRICH, M. The Relationship Between Precision-Recall and ROC Curves. **ICML**, 2006.

[19] SOKOLOVA, M.; LAPALME, G. A Systematic Analysis of Performance Measures for Classification Tasks. **Information Processing & Management**, 2009.

[20] AMIT, D. J.; GUTFREUND, H.; SOMPOLINSKY, H. Spin-glass models of neural networks. **Physical Review A**, 1985.

[21] ENGEL, A.; VAN DEN BROECK, C. **Statistical Mechanics of Learning**. [s.l.] Cambridge University Press, 2001.

[22] HOPFIELD, J. J. Neural networks and physical systems with emergent collective computational abilities. **Proceedings of the National Academy of Sciences**, v. 79, n. 8, p. 2554–2558, 1982.

[23] MÉZARD, M.; PARISI, G.; VIRASORO, M. A. **Spin Glass Theory and Beyond**. [s.l.] World Scientific, [s.d.].

[24] CHOROMANSKA, A. et al. The Loss Surfaces of Multilayer Networks. **arXiv / MLR proceedings**, 2015.

[25] BELKIN, M.; HODA, S. Reconciling modern machine-learning practice and the classical bias–variance trade-off (double descent). **PNAS / arXiv**, 2019.

[26] SHERRINGTON, D.; KIRKPATRICK, S. Solvable model of a spin-glass. **Physical Review Letters**, v. 35, n. 26, p. 1792–1796, 1975.

[27] AMAZON WEB SERVICES. **O que é visão computacional?** Disponível em: <https://aws.amazon.com/pt/what-is/computer-vision>.

[28] DATACAMP. **O que é visão computacional?** Disponível em: <https://www.datacamp.com/pt/blog/what-is-computer-vision>.

[29] SALK INSTITUTE. **Como o cérebro reconhece o que o olho vê**. Disponível em: <https://www.salk.edu/pt/comunicado-de-imprensa/cerebro-reconhece-olho-ve>.

[30] VISÃO COMPUTACIONAL BRASIL. **Identificação, detecção, reconhecimento e segmentação de imagem e objetos**. Disponível em: <https://visaocomputacional.com.br/identificacao-deteccao-reconhecimento-e-segmentacao-de-imagem-e-objetos>.

[31] KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. ImageNet Classification with Deep Convolutional Neural Networks. **Advances in Neural Information Processing Systems (NeurIPS)**, 2012.

[32] MATHWORKS. **yolov2ObjectDetector Documentation**. Disponível em: <https://www.mathworks.com/help/vision/ref/yolov2objectdetector.html>.

[33] V7 LABS. **YOLO Object Detection Explained**. Disponível em: <https://www.v7labs.com/blog/yolo-object-detection>.

[34] SONI, S. **YOLO v2 Comprehensive Tutorial**. Disponível em: <https://medium.com/@sachinsoni600517/yolo-v2-comprehensive-tutorial-building-on-yolo-v1-mistakes-aa7912292c1a>.