Educação Matemática em Revista – RS

SBEM-RS
Publicação da Sociedade Brasileira de Educação Matemática – Regional Rio Grande do Sul

eISSN 3085-8771 • DOI https://doi.org/10.37001/EMR-RS-v.2-n.27-2026.4706 • v. 2, n. 27 • 2026 • Artigo de Pesquisa

A aplicação do cálculo diferencial e integral em visão computacional: detecção de objetos em imagens

The application of differential and integral calculus in computer vision: object detection in images

La aplicación del cálculo diferencial y integral en la visión por computadora: detección en imágenes

Karolliny Danielle Santos 0000-0003-4537-2919
Universidade do Estado de Minas Gerais, Minas Gerais, Brasil. E-mail: karollinny.ds@gmail.com

Brenda Abreu Almeida 0009-0001-4216-1983
Universidade do Estado de Minas Gerais, Minas Gerais, Brasil. E-mail: brendaalmeidxx@gmail.com

Caio Henrique Braga Teixeira 0009-0005-1323-0168
Universidade do Estado de Minas Gerais, Minas Gerais, Brasil. E-mail: caiohbragateixeira@gmail.com

Guilherme Tadeu Almeida Sardinha Marques 0009-0008-0141-8851
Universidade do Estado de Minas Gerais, Minas Gerais, Brasil. E-mail: guisardinha09@gmail.com

Héber Stavrakas Gaipo 0009-0003-3245-3004
Universidade do Estado de Minas Gerais, Minas Gerais, Brasil. E-mail: contatohebergaipo@gmail.com

Pedro Henrique Pereira 0009-0007-5339-7899
Universidade do Estado de Minas Gerais, Minas Gerais, Brasil. E-mail: pedrohenriquepereira02@gmail.com

Resumo: Este artigo explora aplicações do cálculo diferencial e integral na visão computacional, focando na detecção de objetos em imagens. Utilizando a arquitetura MobileNet SSD, o trabalho demonstra como conceitos de cálculo, como derivadas, integrais e otimização, são fundamentais para o desenvolvimento e aplicação de algoritmos de detecção de imagens. A metodologia, classificada como pesquisa aplicada com abordagem de pesquisa-ação, envolveu revisão bibliográfica, seleção do modelo, implementação computacional do código e análise dos resultados. Os experimentos demonstraram a eficiência da MobileNet SSD na detecção de objetos em imagens, com um tempo de processamento comparável ao tempo de reação humano. Conclui-se que a compreensão do cálculo subjacente é crucial para o desenvolvimento de soluções inovadoras e eficientes em visão computacional.

Palavras-chave: Visão Computacional. Cálculo Diferencial e Integral. MobileNet SSD. Detecção de Objetos. Inteligência Artificial.

Abstract: This article explores applications of differential and integral calculus in computer vision, focusing on object detection in images. Using the MobileNet SSD architecture, the study demonstrates how calculus concepts such as derivatives, integrals, and optimization are fundamental to the development and implementation of image detection algorithms. The methodology, characterized as applied research with an action-research approach, involved a literature review, model selection, computational implementation of the code, and analysis of the results. The experiments demonstrated the efficiency of MobileNet SSD in object detection, processing times comparable to human reaction times. It is concluded that the underlying calculus is crucial for developing innovative and efficient solutions in computer vision.

Keywords: Computer Vision. Differential and Integral Calculus. MobileNet SSD. Object Detection. Artificial Intelligence.

Resumen: Este artículo explora aplicaciones del cálculo diferencial e integral en la visión por computadora, con énfasis en la detección de objetos en imágenes. Utilizando la arquitectura MobileNet SSD, el trabajo demuestra cómo conceptos de cálculo, como derivadas, integrales y optimización, son fundamentales para el desarrollo y la aplicación de algoritmos de detección de imágenes. La metodología, caracterizada como investigación aplicada

con un enfoque de investigación-acción, incluyó una revisión bibliográfica, la selección del modelo, la implementación computacional del código y el análisis de los resultados. Los experimentos demostraron la eficiencia de MobileNet SSD en la detección de objetos en imágenes, con un tiempo de procesamiento comparable al tiempo de reacción humano. Se concluye que la comprensión del cálculo subyacente es crucial para el desarrollo de soluciones innovadoras y eficientes en visión por computadora.
Palabras clave: Visión por Computador. Cálculo Diferencial e Integral. MobileNet SSD. Detección de Objetos. Inteligencia Artificial.

1 Introdução

O Cálculo Diferencial e Integral é uma disciplina fundamental no ensino superior, essencial para compreender e modelar fenômenos que envolvem variações contínuas e a acumulação de quantidades (Stewart; Kokoska, 2010). Apesar de sua relevância para o desenvolvimento do raciocínio lógico e para o embasamento de inovações tecnológicas, a aprendizagem de Cálculo é frequentemente apontada como um processo desafiador para os estudantes universitários, exigindo novas estratégias que superem as práticas tradicionais de ensino (Richit; Richit; Teilor, 2023).

A Visão Computacional, área da Inteligência Artificial que busca emular a visão humana, surge como uma ferramenta poderosa para gerar conhecimento a partir de dados visuais. Nesse contexto, a detecção de objetos utilizando Redes Neurais Convolucionais (CNNs) desponta como uma aplicação essencial (Feng *et al*., 2019). O uso dessas tecnologias abrange setores variados, viabilizando a segmentação semântica em carros autônomos (Trindade, 2018), garantindo precisão no monitoramento de segurança robótica (Carvalho *et al*., 2013), modernizando a pulverização agrícola por meio de drones (Daniel; Dalbianco, 2023) e aumentando a taxa de cura na detecção precoce de tumores na medicina (Katzman, 2018).

O treinamento e o funcionamento das CNNs demonstram claramente a aplicação prática do cálculo. As redes operam aplicando filtros convolucionais, uma operação que envolve a integral do produto de duas funções, representando o filtro e a imagem (Li *et al*., 2018). Além disso, a otimização da função de custo, responsável por medir o erro da rede, utiliza derivadas e algoritmos baseados no gradiente descendente para ajustar iterativamente os pesos dos neurônios durante a retropropagação.

Este artigo investiga a relação entre a Visão Computacional e o Cálculo, mostrando como conceitos matemáticos aparentemente abstratos são a base para o funcionamento dos algoritmos de detecção de objetos. Assim, visa responder o quão necessário é o domínio de diversos conceitos de cálculo para as engenharias e principalmente para o desenvolvimento de algoritmos desse tipo.

A lacuna que este estudo busca preencher reside na necessidade de explicitar a conexão entre a teoria matemática e a aplicação prática, voltada para a programação em visão computacional. Mediante uma abordagem metodológica que combina revisão bibliográfica e implementação de código-fonte, demonstra-se como conceitos matemáticos, como derivadas, integrais e otimização, são aplicados na detecção de objetos.

2 Materiais e Métodos

Este trabalho foi classificado como uma pesquisa aplicada. Conforme Prodanov e Freitas (2013), o principal objetivo desse tipo de pesquisa é gerar conhecimento que possa ser utilizado na solução de problemas específicos. O estudo propôs uma metodologia aplicada para demonstrar a aplicação do cálculo na detecção de objetos em imagens. A metodologia compreendeu a revisão bibliográfica, a seleção do modelo computacional utilizado e a aplicação do algoritmo, detalhado na seção de desenvolvimento. Para isso, foi adotada a abordagem de pesquisa-ação, um método que envolve a participação ativa na busca por soluções voltadas a um problema ou necessidade coletiva (Prodanov; Freitas, 2013).

Para a aplicação prática deste estudo, adotou-se o modelo MobileNet SSD (*Single Shot MultiBox Detector*), amplamente reconhecido por equilibrar precisão e alta velocidade de inferência. Trata-se de uma arquitetura projetada para operar de maneira eficiente em ambientes com recursos computacionais limitados. O modelo utiliza a rede MobileNet (Howard *et al*., 2017) como espinha dorsal (*backbone*) para a extração de características, otimizando o processamento por meio de convoluções separáveis em profundidade (*depthwise separable convolutions*), que reduzem drasticamente o custo computacional e o número de parâmetros em comparação às convoluções tradicionais. Essa estrutura base é associada ao framework SSD (Liu *et al*., 2016), que tem a capacidade de prever caixas delimitadoras (*bounding boxes*) e classificar objetos de diferentes escalas em uma única passagem pela rede (*single shot*), dispensando etapas custosas de geração de propostas de região.

Utilizou-se o modelo MobileNet SSD justamente devido a essa alta eficiência computacional, o que o torna apto para rodar em diversas configurações de computadores, desde os sistemas mais ultrapassados até os mais sofisticados. Além disso, por ser vastamente documentado na literatura científica, o modelo facilita a implementação, os testes e a compreensão do fluxo algorítmico. A implementação computacional foi desenvolvida na linguagem Python, valendo-se da biblioteca *OpenCV* para as operações e filtros matemáticos de processamento de

imagens, e do submódulo *dnn (Deep Neural Network*) do próprio *OpenCV*, que serve como interface direta de leitura e execução da arquitetura MobileNet SSD.

3 Desenvolvimento

Matematicamente, a convolução discreta 2D entre uma imagem I(x, y) e um filtro K(i, j) de tamanho (2n + 1)∗(2m + 1), onde n, m são inteiros positivos que definem o tamanho do filtro (kernel) e a quantidade de pixel, em cada direção, é definida como:

$$S(x,y)= \sum_{i=-n}^{n} \sum_{j=-m}^{m} I(x-i, y-j) \cdot K(i,j) \quad (1)$$

Onde S(x, y) representa o valor do pixel de saída na posição (x, y) após a aplicação do filtro, resultado da convolução naquela coordenada específica da imagem.

I(x – i, y – j) representa o valor do pixel de entrada na posição (x − i, y − j) e I é a função que descreve a imagem de entrada. Ela mapeia a coordenada (x, y) da imagem para um valor de intensidade (por exemplo, brilho em escala de cinza ou valores RGB para imagens coloridas). x e y são as coordenadas do pixel central da região da imagem que está sendo processada pelo filtro e i e j são os deslocamentos em relação ao pixel central (Szeliski, 2022).

K(i, j) representa o valor do filtro kernel na posição (i, j), onde K é a função que descreve o filtro. Ela define os pesos que serão aplicados aos pixels vizinhos durante a convolução e i e j são as coordenadas no filtro.

∑ representa o somatório e indica que os valores nos limites especificados devem ser somados, sendo esses valores i = − n até n e j = − m até m. O filtro tem tamanho de (2n + 1) linhas e (2m + 1) colunas. A fórmula (2n + 1)∗(2m + 1) garante que o tamanho do kernel em cada dimensão seja sempre um número ímpar, permitindo que o kernel tenha um pixel central bem definido, o que auxilia na aplicação da convolução. Kernels maiores capturam informações de uma área maior da imagem, enquanto kernels menores focam em detalhes mais localizados, para isso a escolha dos valores de n e m dependem da aplicação e das características que se deseja extrair da imagem (Gonzalez; Woods, 2006).

Diversos filtros podem ser aplicados para diferentes propósitos durante o pré-processamento das imagens, cada um projetado para realçar ou suprimir certas características da imagem (Szeliski, 2022). A aplicação de um filtro envolve a operação matemática de convolução, que relaciona diretamente a integral do produto de duas funções. Alguns exemplos da utilização do cálculo nessa implementação, conforme fundamentam Gonzalez e Woods (2006) e Stewart e Kokoska (2010) incluem:

• Filtro Gaussiano (Suavização): Utiliza uma função gaussiana, definida por exponenciais e conceitos de desvio padrão (estatística, intimamente ligada à probabilidade e ao cálculo), para suavizar a imagem, reduzindo ruído e detalhes finos. A função gaussiana é definida por uma integral.

• Filtro Laplaciano (Detecção de Bordas): Aproxima a segunda derivada da imagem, realçando regiões com mudanças abruptas de intensidade, que correspondem às bordas. O filtro Laplaciano é um exemplo de como o cálculo diferencial é aplicado diretamente no processamento de imagens.

• Filtro Sobel (Detecção de Bordas Direcionais): Calcula o gradiente da imagem em direções específicas (horizontal e vertical), permitindo detectar bordas com diferentes orientações. O conceito de gradiente vem do cálculo vetorial, parte integrante do cálculo multivariável.

A MobileNet SSD é uma arquitetura de rede neural convolucional projetada para detecção de objetos em tempo real, mesmo em dispositivos com recursos computacionais limitados (Li *et al.,* 2018). Ela combina a arquitetura MobileNet (Howard *et al.,* 2017), eficiente em termos de computação, com o algoritmo *Single Shot MultiBox Detector* – SSD (Liu *et al.,* 2016), que permite a detecção de múltiplos objetos em uma única passagem (inferência) pela rede.

4 Resultados e discussão

O presente trabalho apresenta a implementação do código mostrando aplicações do cálculo diferencial e integral em um modelo de detecção de objetos. Foram utilizados os seguintes passos: carregamento do modelo pré-treinado (arquivos prototxt e caffemodel), pré-processamento da imagem de entrada (redimensionamento e normalização), execução da detecção de objetos e pós-processamento dos resultados (filtragem por confiança e desenho das *bounding boxes* na imagem). Os trechos de código a seguir demonstram a execução do programa, que utiliza a arquitetura MobileNet SSD para gerar como saída a imagem esperada e apresenta a aplicabilidade do cálculo para cada passo da implementação código:

Figura 1 – Variáveis utilizadas no programa

VARIAVEIS:

- `CLASSES`: Lista de rótulos de classe que o MobileNet SSD foi treinado para detectar.
- `CORES`: Matriz de cores aleatórias para cada classe.
- `rede`: Modelo de rede neural carregado.
- `imagem`: Imagem de entrada carregada.
- `altura, largura`: Dimensões da imagem de entrada.
- `blob`: Objeto blob de entrada para a rede neural.
- `deteccoes`: Detecções e previsões da rede neural.
- `confianca`: Confiança da detecção.
- `indice_classe`: Índice do rótulo de classe da detecção.
- `caixa_delimitadora`: Coordenadas da caixa delimitadora da detecção.
- `inicioX, inicioY, fimX, fimY`: Coordenadas inteiras da caixa delimitadora.
- `rotulo`: Rótulo de classe formatado com a confiança.
- `y`: Posição vertical do rótulo de texto.

Fonte: Elaborada pelos autores (2025)

Na Figura 2 são declaradas todas as classes com as quais o modelo foi treinado e a gama de cores a ser utilizada na decoração de cada uma das caixas delimitadoras de objetos.

Figura 2 – Inicializando classes e cores

```
// Inicialização das classes e cores
CLASSES ← ["background", "aeroplane", "bicycle", "bird", "boat", "bottle",
"bus", "car", "cat", "chair", "cow", "diningtable", "dog", "horse",
"motorbike", "person", "pottedplant", "sheep", "sofa", "train",
"tvmonitor"] 1
CORES ← gerar_cores_aleatorias(comprimento(CLASSES), 0, 255)
```

Fonte: Elaborada pelos autores (2025)

Na Figura 3 ocorre o pré-processamento da imagem (redimensionamento e normalização), antes de alimentar a rede neural.

Figura 3 – Pré-processamento de imagem (redimensionamento e normalização)

```
// Carregar e pré-processar a imagem
imagem ← carregar_imagem(caminho_imagem)
altura, largura ← obter_dimensoes(imagem)
blob ← criar_blob_imagem(redimensionar_imagem(imagem, 300, 300), 0.007843,
(300, 300), 127.5)
```

Fonte: Elaborada pelos autores (2025)

4.1 Redimensionamento e Interpolação

Redimensionar uma imagem significa alterar suas dimensões (largura e altura), aumentando ou diminuindo o número de pixels. Quando uma imagem é ampliada, novos pixels precisam ser criados e quando é reduzida, alguns pixels precisam ser descartados. A interpolação é o método usado para determinar o valor desses novos pixels quando ampliada ou para combinar os valores dos pixels existentes ao reduzir a imagem, função mostrada na Figura 3. Seguem alguns tipos de interpolação:

• Interpolação como Aproximação de Função: Uma imagem pode ser analisada como uma função matemática discreta, onde cada pixel (x, y) possui um valor de intensidade f(x, y). Redimensionar a imagem é como mostrar essa função em novos pontos. A interpolação, então, age como uma forma de aproximar o valor da função nesses novos pontos, com base nos valores conhecidos dos pixels originais (Gonzalez; Woods, 2006).

• Métodos de Interpolação e Cálculo: Existem diversos métodos de interpolação, cada um com diferentes graus de complexidade e precisão. Alguns exemplos comuns incluem:

– Interpolação Vizinho Mais Próximo: Simplesmente copia o valor do pixel original mais próximo. Não envolve cálculos complexos, mas pode resultar em imagens pixelizadas (Szeliski, 2022).

– Interpolação Bilinear: Calcula a média ponderada dos quatro pixels vizinhos. Envolve operações aritméticas e ponderação, sendo conceitos básicos do cálculo (Szeliski, 2022).

– Interpolação Bicúbica: Utiliza um polinômio de terceiro grau para aproximar a função e calcular o valor do novo pixel. Este método, mais sofisticado, produz resultados mais suaves e se baseia diretamente em conceitos de cálculo, como derivadas e polinômios (Szeliski, 2022).

4.2 Normalização e Operações Aritméticas

A normalização, neste contexto, refere-se ao processo de ajustar os valores dos pixels para uma determinada faixa. Na Figura 3, pode-se ver uma parte do código onde a função para criar *blob* na imagem realiza a normalização subtraindo 127,5 de cada valor de pixel e multiplicando o resultado por 0,007843. Isso efetivamente escala os valores dos pixels para uma faixa aproximadamente entre [−1, 1].

Cálculo e Operações Básicas: Embora a normalização neste caso envolve apenas operações aritméticas simples (subtração e multiplicação), ela é fundamental para o bom funcionamento das redes neurais. A normalização ajuda a evitar problemas de instabilidade numérica

durante o treinamento e melhora o desempenho da rede. Além disso, o conceito de escala e transformação de dados é relevante em diversos contextos de cálculo.

Na Figura 4, o algoritmo obtém as detecções. Internamente, a rede neural realiza inúmeras operações, incluindo convoluções (integrais), ativações não-lineares (funções deriváveis) e, no treinamento, o processo de *backpropagation* se baseia fortemente em derivadas parciais para calcular gradientes para a otimização da rede.

Figura 4 – Obtendo as detecções

```
// Detecção de objetos
IMPRIMIR "[INFO] computando detecções de objetos..."
definir_entrada_rede(rede, blob)
deteccoes ← propagacao_para_frente_rede(rede)
```

Fonte: Elaborada pelos autores (2025)

4.3 Convoluções (Integrais)

São as camadas convolucionais, o coração das redes neurais convolucionais (CNNs). Elas aplicam filtros (*kernels*) à imagem de entrada para extrair características relevantes, como bordas, texturas e padrões (Li *et al.*, 2018), passo apresentado na Figura 4. A operação de convolução, matematicamente, é uma forma discreta de integral, como descrito na fundamentação teórica, na equação (1).

O cálculo na prática: o filtro percorre o domínio da imagem, multiplicando seus valores pelos valores dos pixels correspondentes e somando os resultados. Essa soma ponderada é equivalente a calcular a integral do produto das duas funções (filtro e imagem) em uma região específica. Embora a implementação computacional utilize somatórios, a base matemática subjacente é a integração.

4.4 Ativações Não-Lineares (Funções Deriváveis)

Após a convolução, uma função de ativação não-linear é aplicada ao resultado. Essa função introduz não-linearidade no modelo, permitindo que a rede aprenda padrões complexos que não podem ser representados por funções lineares. Cálculo na prática: as funções de ativação devem ser deriváveis para que o processo de treinamento, baseado em gradiente, funcione corretamente. Exemplos comuns de funções de ativação incluem a função sigmoide, a função tangente hiperbólica e a ReLU (*Rectified Linear Unit*). O cálculo diferencial é essencial para calcular as derivadas dessas funções.

• Por que deriváveis?

A derivada da função de ativação é usada no processo de *backpropagation* para calcular o gradiente da função de custo em relação aos pesos da rede.

4.5 Backpropagation e Derivadas Parciais (Gradientes e Otimização)

Backpropagation é a parte central do algoritmo no treinamento de redes neurais. Ele calcula o gradiente da função de custo (que mede o erro da rede) em relação aos pesos da rede. Esse gradiente indica a direção na qual os pesos devem ser ajustados para minimizar o erro (Lillicrap *et al.*, 2020). O *backpropagation* utiliza a regra da cadeia do cálculo diferencial para calcular as derivadas parciais da função de custo em relação a cada peso da rede. Essas derivadas parciais formam o gradiente.

O gradiente descendente é usado por algoritmos de otimização para atualizar os pesos da rede iterativamente. O gradiente "desce" a superfície da função de custo, buscando o ponto de mínimo, que representa o erro mínimo da rede.

As convoluções (baseadas em integrais), as ativações não-lineares (funções deriváveis) e o *backpropagation* (que usa derivadas parciais para calcular gradientes) são operações fundamentais nas redes neurais e dependem fortemente de conceitos de cálculo. Embora as bibliotecas de aprendizado de máquina abstraem a complexidade matemática, entender os fundamentos do cálculo é crucial para compreender o funcionamento e o treinamento das redes neurais.

Na Figura 5, um loop itera sobre as detecções que produzem geralmente as coordenadas da *bounding box* (caixa delimitadora) em um formato normalizado, relativo ao tamanho da imagem de entrada da rede neural. Para desenhar a *bounding box* na imagem original, que pode ter dimensões diferentes, é necessário transformar essas coordenadas normalizadas para as coordenadas da imagem original. Esse processo envolve conceitos de geometria analítica e álgebra linear, intimamente relacionados ao cálculo.

Figura 5 – Loop sobre as detecções

```
// Processar as detecções
PARA cada detecção em deteccoes FAZER
confianca ← obter_confianca_deteccao(deteccao)

SE confianca > confianca_minima ENTAO
indice_classe ← obter_indice_classe(deteccao)
caixa_delimitadora ← obter_caixa_delimitadora(deteccao, largura, altura)
inicioX, inicioY, fimX, fimY ←
converter_caixa_para_inteiros(caixa_delimitadora)
```

Fonte: Elaborada pelos autores (2025)

• As coordenadas x e y do centro da *bounding box*, normalizadas para o intervalo [0,1], onde 0 representa a extremidade esquerda/superior da imagem e 1 representa a extremidade direita/inferior.

• width_norm e height_norm são a largura e altura da *bounding box*, também normalizadas para o intervalo [0,1].

4.6 Transformação para Coordenadas da Imagem Original

Para desenhar a *bounding box* na imagem original, com largura w e altura h, as coordenadas normalizadas precisam ser transformadas:

$$x = x_{norm} \cdot w \quad (2)$$

$$y = y_{norm} \cdot h \quad (3)$$

$$width = width_{norm} \cdot w \quad (4)$$

$$height = height_{norm} \cdot h \quad (5)$$

A partir dessas informações, as coordenadas dos cantos superior esquerdo (inicioX, inicioY) e inferior direito (fimX, fimY) da *bounding box,* podem ser calculadas:

$$inicioX = \int \left(x - \frac{width}{2}\right) \quad (6)$$

$$inicioY = \int \left(y - \frac{height}{2}\right) \quad (7)$$

$$fimX = \int \left(x + \frac{width}{2}\right) \quad (8)$$

$$fimY = \int \left(y + \frac{height}{2}\right) \quad (9)$$

A conversão para inteiros (int()) é necessária porque as coordenadas dos pixels devem ser números inteiros.

A etapa de transformação para as coordenadas da imagem original está representada na Figura 5.

4.7 Conexão com Geometria Analítica e Álgebra Linear

• Escalonamento: A multiplicação das coordenadas normalizadas pela largura e altura da imagem original é uma forma de escalonamento, um conceito fundamental em geometria analítica e álgebra linear. É equivalente a multiplicar um vetor pelas dimensões da imagem, representadas como uma matriz diagonal.

• Translação: Calcular as coordenadas dos cantos da bounding box a partir do centro e das dimensões envolve translações, outro conceito importante em geometria analítica.

• Representação Matricial: Todo esse processo de transformação pode ser representado de forma mais compacta e eficiente usando multiplicação de matrizes, um elemento central da álgebra linear. As coordenadas normalizadas e as dimensões da imagem podem ser representadas como vetores e matrizes, e a transformação pode ser realizada com uma única multiplicação de matrizes. Na Figura 6 o modelo é configurado para exibir na tela as informações.

Figura 6 – Exibindo as informações na tela

```
    rotulo ← formatar_rotulo(CLASSES[indice_classe], confianca * 100)
    IMPRIMIR "[INFO] " + rotulo

    desenhar_retangulo(imagem, inicioX, inicioY, fimX, fimY,
CORES[indice_classe])
    y ← inicioY - 15 SE inicioY - 15 > 15 SENAO inicioY + 15
    desenhar_texto(imagem, rotulo, inicioX, y, CORES[indice_classe])

FIM SE
FIM PARA
```

Fonte: Elaborada pelos autores (2025)

• Probabilidade: O *rotulo* estabelece o formato de texto que vai imprimir a informação, sendo essas informações probabilísticas com duas casas decimais, sendo a informação capturada no escopo de classes através do índice passado pelo algoritmo e a probabilidade sendo o valor multiplicado por cem, para aparecer no formato 100,00%. Essa variável *rotulo* é impressa no prompt de comando.

Gera-se na imagem o retângulo que vai delimitar o objeto, coordenadas de início e fim do retângulo passadas por (inicioX, inicioY) e (fimX, fimY), a cor do retângulo definida por *CORES[indice_classe],* apresentado na Figura 2.

• A variável y armazena a coordenada final onde o texto de informação vai ser inserido com o retângulo que delimita o objeto.

• Então o programa insere o texto *rotulo* na imagem nas coordenadas (inicioX, y) com a fonte na mesma cor do retângulo.

• A biblioteca *cv2* gera um arquivo com essa nova imagem que contém essas informações.

O programa é encerrado através das linhas de código presentes na Figura 7. É exibida na tela a imagem gerada pelo programa e o programa aguarda que qualquer tecla seja pressionada, concluindo de vez a execução do programa.

Embora o código em si não mostra os cálculos complexos diretamente, ele se baseia em bibliotecas e algoritmos fundamentados em conceitos de cálculo. A própria arquitetura da rede

neural convolucional, MobileNet SSD, é construída com base em operações como convolução (sendo a integral) e utiliza cálculo diferencial para o processo de treinamento (*backpropagation* com gradiente descendente).

Figura 7 – Encerrando o programa

```
// Salvar e exibir a imagem de saída
salvar_imagem("images/output/image.jpg", imagem)
exibir_imagem("Output", imagem)
esperar_tecla()
```

Fonte: Elaborada pelos autores (2025)

A manipulação de imagens, mesmo em etapas que parecem simples como redimensionamento, pode envolver interpolações que, em sua essência, são aproximações de funções, um conceito central no cálculo. A detecção de bordas, muitas vezes um pré-processamento para detecção de objetos, usa derivadas para encontrar mudanças abruptas nos valores dos pixels. Portanto, mesmo que o código não realize integrais e derivadas explicitamente, ele se beneficia da teoria e das ferramentas fornecidas pelo cálculo para realizar a tarefa de detecção de objetos.

4.8 Experimentos

Os testes computacionais foram realizados em um computador Dell Inspiron 15 3530 com processador Intel(R) Core(TM) i5-1335U de 13ª geração e 1,30 GHz, com 16GB de memória RAM, SSD de 512 GB e sistema operacional de 64 bits.

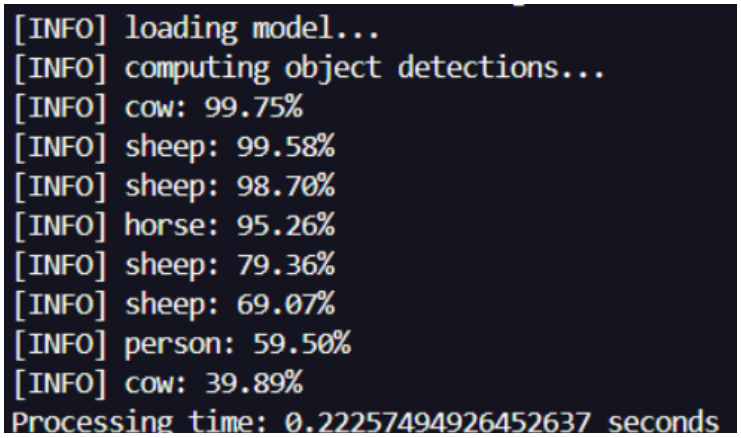
Figura 8 – Resultados

Fonte: Elaborada pelos autores (2025)

Para verificar a eficiência do modelo implementado, foi analisada uma figura na qual é apresentada na Figura 8. Foi adicionada a figura ao código e ele realizará a detecção de quais objetos aparecem na figura. Como arquivo de entrada temos a Figura 8 (a) e de saída temos a Figura 8 (b), mostrando em detalhes os animais que foram encontrados na imagem dada.

Como mostra a Figura 9, a máquina 6 levou cerca de 0,222 milissegundos para realizar todo o procedimento de detecção de objetos da Figura 8, mostrando a acurácia de cada objeto detectado. Dada a importância da utilização do recurso, principalmente quando comparado com dados públicos como o gráfico disponibilizado pela Human Benchmark. Martinsons (1994) mostra em sua pesquisa que o tempo de reação das pessoas que participaram do teste realizado pela empresa, que consistia em medir o tempo que uma pessoa leva para clicar no mouse após uma mudança de cor e, de acordo com a página de estatísticas, o tempo de reação médio das pessoas é de 0,284 milissegundos. Logo, pode-se notar que a implementação de detecção de objetos neste trabalho, mostrou-se eficiente. O processo em estudo, consiste em:

Figura 9 – Tempo de processamento



Fonte: Elaborada pelos autores (2025)

A imagem que será analisada é passada como parâmetro no prompt para o algoritmo.

As classes em que o modelo foi treinado são carregadas e é atribuída inicialmente uma cor aleatória para cada classe para servir de ilustração na saída do modelo (Figura 2).

O modelo é carregado pela aplicação utilizando como parâmetro a arquitetura e a configuração dos pesos dentro da estrutura.

Ocorre o pré-processamento da imagem, sendo o redimensionamento do tamanho original para 300x300 píxels e a normalização dos valores de cada píxel para o intervalo entre 0 e 1, o que facilita as operações feitas nos passos seguintes (Figura 3).

Quando a imagem entra na rede é aplicado um filtro a ela, para permitir ao modelo computar informações relevantes dentro da imagem, como bordas, padrões complexos e comparar com a base de dados feita no treinamento do algoritmo (Figura 4).

O processo de *backpropagation* é iniciado forçando a imagem percorrer a rede diversas vezes enquanto os pesos são ajustados, buscando encontrar no banco de dados o padrão gráfico de imagem resposta que possua pontos com a menor distância entre o gráfico formado pela imagem de entrada e encontrar a resposta que melhor se pareça com o padrão detectado. Essa etapa é responsável por permitir ao algoritmo detectar diversos objetos em um mesmo processo de detecção (Figura 4).

A otimização do algoritmo é feita por derivadas parciais visando direcionar os pesos sempre na direção de reduzir a distância entre o gráfico da imagem de entrada e os gráficos das possíveis respostas no banco de dados e encontrar o mínimo da função (Figura 4).

Após essa comparação, são consideradas como detecções as classes da base de dados que se aproximam do padrão percebido na imagem de entrada (Figura 5).

O argumento de confiança filtra detecções para cada um dos objetos percebidos na imagem que estejam dentro do intervalo aceitável pré-estabelecido (entre 0 e 1, sendo 0 e 100%) (Figura 5).

Um loop itera sobre todas as detecções aceitas e produz as coordenadas das caixas delimitadoras a serem desenhadas na imagem final, processo feito com base na imagem normalizada que precisa transformar essas coordenadas com base em geometria analítica e álgebra linear (escalonamento, translação e matriz) (Figura 5).

Então a manipulação é feita para que os desenhos apareçam na imagem e os resultados sejam exibidos, tanto no console, quanto na imagem que é aberta e salva na memória local (Figura 6).

Portanto, é notório que o uso do cálculo diferencial e integral é parte fundamental para que a tecnologia funcione. As derivadas e as integrais são excelentes aliadas ao bom funcionamento do modelo de detecção de objetos, que por sua vez é um ótimo aliado dentro dos setores da agricultura, medicina, categorização de objetos e transporte. O algoritmo se mostra adepto pela sua eficiência, assertividade e resultados satisfatórios.

Pode-se notar pela aplicação na detecção de objetos que, o cálculo diferencial e integral desempenha um papel fundamental no treinamento e operação de redes neurais convolucionais, assim sendo essencial o domínio pleno desses conceitos (derivada, geometria analítica e álgebra

linear e integral) ao desenvolver softwares que envolvam aprendizagem de máquina para análise de dados em imagens. Mesmo que muitas vezes seja obscurecido pelas bibliotecas e frameworks de alto nível.

5 Conclusão

Este artigo demonstrou a intrínseca relação entre o cálculo diferencial e integral e a visão computacional, atendendo ao objetivo principal de destacar a importância do cálculo no desenvolvimento e aplicação de algoritmos de detecção de objetos. A análise do código, utilizando a MobileNet SSD, evidenciou como conceitos como derivadas, integrais e otimização são aplicados em operações como convolução, ativações não-lineares e *backpropagation*.

A discussão sobre interpolação para redimensionamento de imagens e a normalização de dados reforçou ainda mais essa conexão. Os resultados práticos obtidos com a detecção de objetos em imagens e a medição do tempo de processamento, que se mostrou comparável ao tempo de reação humano médio, validam a eficiência da MobileNet SSD e a sua aplicabilidade em diversos setores, como agricultura, medicina e transporte, conforme explorado na fundamentação teórica.

Apesar da abstração proporcionada pelas bibliotecas, o conhecimento do cálculo subjacente se mostrou essencial para a compreensão e o desenvolvimento de soluções mais eficazes em visão computacional. Uma limitação deste estudo reside no foco em uma única arquitetura de rede neural. Pesquisas futuras poderiam explorar a aplicação do cálculo em outras arquiteturas e em diferentes estágios do pipeline de visão computacional, como no pré-processamento de imagens e na interpretação dos resultados.

Referências

CARVALHO, G.; OLIVEIRA, J. de; SILVA, E. da; NETTO, S.; FREITAS, G.; MOTTA-RIBEIRO, G.; COSTA, R. Um sistema de monitoramento para detecção de objetos em tempo real empregando câmera em movimento**. In: SIMPÓSIO BRASILEIRO DE TELECOMUNICAÇÕES**, 31., 2013, Fortaleza. Anais [...]. Fortaleza: Sociedade Brasileira de Telecomunicações, 2013.

DANIEL, D.; DALBIANCO, A. Tecnologia de pulverização com drones: panorama, oportunidades, perspectivas futuras e desafios na agricultura moderna. **Revista Plantio Direto**, v. 2023, n. 193, p. 69–77, 2023.

FENG, X.; JIANG, Y.; YANG, X.; DU, M.; LI, X. Computer vision algorithms and hardware implementations: A survey**. Integration**, v. 69, p. 309–320, 2019. ISSN 0167-9260.

GONZALEZ, R.; WOODS, R. **Digital Image Processing (3rd Edition)**. USA: Prentice-Hall, Inc., 2006. ISBN 013168728X.

HOWARD, A.; ZHU, M.; CHEN, B.; KALENICHENKO, D. **MobileNets: Efficient convolutional neural networks for mobile vision applications**. arXiv preprint arXiv:1704.04861, 2017.

KATZMAN, J. L. et al. Deepsurv: personalized treatment recommender system using a cox proportional hazards deep neural network. **BMC Medical Research Methodology**, v. 18, n. 1, p. 24, 2018. ISSN 1471-2288.

LI, Y.; HUANG, H.; XIE, Q.; YAO, L.; CHEN, Q. Research on a surface defect detection algorithm based on mobilenet-ssd. **Applied Sciences**, v. 8, n. 9, 2018. ISSN 2076-3417.

LILLICRAP, T.; SANTORO, A.; MARRIS, L.; AKERMAN, C.; HINTON, G. Backpropagation and the brain. **Nature Reviews Neuroscience**, v. 21, 2020. ISSN 1471-0048.

LIU, W.; ANGUELOV, D.; ERHAN, D.; SZEGEDY, C.; REED, S.; FU, C.-Y.; BERG, A**.** SSD: Single shot MultiBox detector. **European Conference on Computer Vision**, Springer, Cham, p. 21-37, 2016.

MARTINSONS, M. Benchmarking human resource information systems in Canada and Hong Kong. **Information & Management**, v. 26, n. 6, p. 305-316, 1994.

PRODANOV, C.; FREITAS, E. **Metodologia do trabalho científico: métodos e técnicas da pesquisa e do trabalho acadêmico-2ª Edição**. [S.l.]: Editora Feevale, 2013.

RICHIT, A.; RICHIT, L.; TEILOR, B. Abordagem de máximos e mínimos em um curso universitário de cálculo. **Bolema**, v. 37, n. 77, p. 1036-1062, 2023. Disponível em: https://doi.org/10.1590/1980-4415v37n77a06.

STEWART, J.; KOKOSKA, S. **Calculus: Concepts and contexts**. Toronto: Cengage Learning, 2010.

SZELISKI, R. **Computer vision: algorithms and applications.** 2. ed. [S. l.]: Springer Nature, 2022.

TRINDADE, L. **Detecção de obstáculos para carros autônomos utilizando aprendizado profundo**. 87 p. Trabalho de Conclusão de Curso — Universidade Federal de Santa Catarina, Florianópolis, SC, 2018.

Declarações finais obrigatórias

Disponibilidade de dados: Mais informações sobre o programa, implementação do código e como executá-lo estão disponíveis no link: https://github.com/Heber-Stavrakas-Gaipo/object_detection_with_MobileNetSSD.git

Financiamento: Esta pesquisa não recebeu financiamento específico.

Conflito de interesses: Os autores declaram não haver conflito de interesses.

Aprovação ética e consentimento: Não se aplica.

Contribuição de autoria – CRediT: Autora 1: orientação, revisão, supervisão. Autor 2: escrita – revisão. Autor 3: escrita – revisão. Autor 4: escrita – revisão e edição, validação, revisão do algoritmo, metodologia. Autor 5: escrita – edição e revisão, desenvolvimento do algoritmo, busca e análise do referencial teórico. Autor 6: escrita – revisão, validação, metodologia.

Uso de IA generativa: Foi utilizado o modelo de IA generativa Nano Banana 1.0 apenas para geração da Figura 8 de entrada, servindo apenas para imagem autoral com os objetos que se pretendia analisar via algoritmo. Nenhum modelo de IA generativa foi utilizado nos resultados.

Dados de publicação – preenchimento editorial

Recebido em: 03/09/2025

Aprovado em: 06/08/2026

Publicado em: 18/09/2026

Editor(a) responsável: Adriana Richit

Como citar este trabalho: SANTOS, Karolliny Danielle; ALMEIDA, Brenda Abreu; TEIXEIRA, Caio Henrique Braga; MARQUES, Guilherme Tadeu Almeida Sardinha; GAIPO, Héber Stavrakas; PEREIRA, Pedro Henrique. A aplicação do cálculo diferencial e integral em visão computacional: detecção de objetos em imagens. Educação Matemática em Revista – RS, Porto Alegre, v. 2, n. 27, p. 85-100, 2026. Disponível em: https://doi.org/10.37001/EMR-RS-v.2-n.27-2026.4706.

Taxas: a EMR-RS não cobra taxas de submissão, avaliação, publicação, distribuição ou download.