# Relatório do Exercício Programa - Detectar Bananas

**PSI 3472 - Concepção e Implementação de Sistemas Eletrônicos Inteligentes**

Parte 1 - Prof. Hae
Luigi Hideki Tanaka - 11375246
Lucas Harada - 11449492

## Ambiente de desenvolvimento utilizado

O ambiente de desenvolvimento utilizado para o código foi o Google Collabs, e pode ser compilado ao clicar em **ambiente de execução** -> **executar tudo .**
Para a implementação do programa foi utilizado as seguintes bibliotecas: tensorflow.keras.models, tensorflow.keras.layers, tensorflow.keras.regularizers, numpy, tensorflow.keras.optimizers, tensorflow.keras.activations, tensorflow.keras.callbacks, os, tensorflow.keras.datasets, tensorflow.keras.preprocessing.image, matplotlib.pyplot, pandas, tensorflow, inspect, tensorflow.keras.utils, matplotlib.patches e CV2.
Foi necessário instalar o CV2 no ambiente antes da compilação do programa.

## Operação

O professor pode executar o programa rodando as células do notebook em ordem, executando primeiro as células acima e descendo o notebook. Para a execução do programa, é necessário ter baixado os arquivos presentes em banana-detection.zip, do Drive da disciplina.
Há parâmetros que devem ser configurados, sendo eles os caminhos para os arquivos de banana-detection.zip.
Há também arquivos de entrada necessários, que são os arquivos presentes em banana-detection.zip.
Na linha labels_df =
**pd.read_csv('C:/Users/lucas/banana-detection/banana-detection/bananas_train/label.csv')**
Deve substituir o caminho para onde está o label.csv de banana_train em seu computador (coluna de treino).
Em image_files =
**os.listdir('C:/Users/lucas/banana-detection/banana-detection/bananas_train/images')**
deve substituir o caminho para banana_train/images, diretório de imagens de treino em seu computador (imagens de treino).
Em img_path =
**os.path.join('C:/Users/lucas/banana-detection/banana-detection/bananas_train/images', image_file)** deve substituir
**'C:/Users/lucas/banana-detection/banana-detection/bananas_train/images'** pelo caminho de banana_train/images, diretório de imagens de treino em seu computador.
Em **labels_df_val =**
**pd.read_csv('C:/Users/lucas/banana-detection/banana-detection/bananas_val/label.csv')** deve substituir o caminho pelo arquivo label.csv de bananas_val de dados de validação em seu computador.
Em **image_files =**
**os.listdir('C:/Users/lucas/banana-detection/banana-detection/bananas_val/images')**

deve substituir o caminho para a pasta onde está armazenado images de banana_val em seu computador (imagens de teste).
Em **img_path = os.path.join('C:/Users/lucas/banana-detection/banana-detection/bananas_val/images', image_file**) deve substituir **'C:/Users/lucas/banana-detection/banana-detection/bananas_val/images',** pelo caminho de **'bananas_val/images'** em seu computador**,** diretório de imagens de teste de bananas, em seu computador.

O programa também gera arquivos de saída, sendo eles 'modelo_trabalho_HAE_7.32.keras' e 'inception_prelayer1.keras' com ambos retornando o modelo de rede treinado pelo programa. Só um deles é necessário para fazer o loading do modelo, mas decidimos criar o 'modelo_trabalho_HAE_7.32.keras' por quesitos de organização de nome

## Introdução

O presente relatório descreve a implementação de um programa para a detecção de objetos, especificamente a localização de bananas em imagens utilizando técnicas de Deep Learning. O conjunto de dados utilizado para este exercício é composto por 1.000 imagens de treino e 100 imagens de teste, todas com resolução de 256x256 pixels, contendo uma única banana em diferentes posições, tamanhos e rotações. O objetivo principal é desenvolver um modelo capaz de prever as coordenadas do bounding box da banana, com o desempenho avaliado pelo erro médio absoluto (MAE) entre as coordenadas previstas e as verdadeiras.

## Data augmentation

O data augmentation foi uma estratégia fundamental para aumentar a diversidade do conjunto de dados, melhorando assim a capacidade de generalização do modelo. As imagens foram deslocadas aleatoriamente, e as coordenadas do bounding box foram ajustadas em consequência. A função `manual_translation` aplica um deslocamento aleatório e a função `adjust_bounding_boxes` modifica as coordenadas do bounding box conforme a translação.

Essas técnicas de data augmentation permitiram que o modelo aprendesse a reconhecer bananas em uma variedade de condições, contribuindo para um melhor desempenho.

## Arquitetura do modelo

A arquitetura do modelo foi construída com base na GoogLeNet (Inception), uma rede neural conhecida por sua eficiência na extração de características de imagens. Esta arquitetura é particularmente vantajosa para tarefas de detecção de objetos, pois combina múltiplas escalas de convolução, permitindo a captura de informações ricas e complexas de maneira eficaz. O modelo utiliza uma série de camadas convolucionais, cada uma com filtros de diferentes tamanhos, o que possibilita a detecção de padrões variados em diferentes níveis de abstração.

As camadas convolucionais são seguidas por uma série de operações de pooling, que reduzem a dimensionalidade das características extraídas e, ao mesmo tempo, preservam

as informações mais relevantes. A combinação dessas camadas resulta em uma representação profunda da imagem, permitindo ao modelo identificar detalhes pequenos, como a forma da banana, bem como padrões mais amplos que definem seu contexto na imagem.

A saída final da rede é composta por uma camada densa projetada para produzir quatro valores correspondentes às coordenadas do bounding box da banana. Esses quatro valores representam, respectivamente, os limites da caixa: xmin, ymin, xmax e ymax. Essa estrutura é crucial, pois cada imagem resulta em uma saída que consiste em quatro números, permitindo uma detecção precisa da posição e do tamanho da banana na imagem.

Para treinar o modelo, a função de perda foi configurada como o erro médio absoluto (MAE). Essa função calcula a média da diferença absoluta entre as coordenadas previstas e as verdadeiras, o que é essencial para problemas de regressão, como a detecção de bounding boxes. O valor do MAE foi ajustado para refletir a escala original das imagens, multiplicando por 32, já que as imagens foram normalizadas para um tamanho de 32x32 pixels durante o pré-processamento. Essa abordagem garante que o modelo aprenda a minimizar a diferença entre suas previsões e os valores reais, resultando em uma detecção mais precisa dos objetos nas imagens.

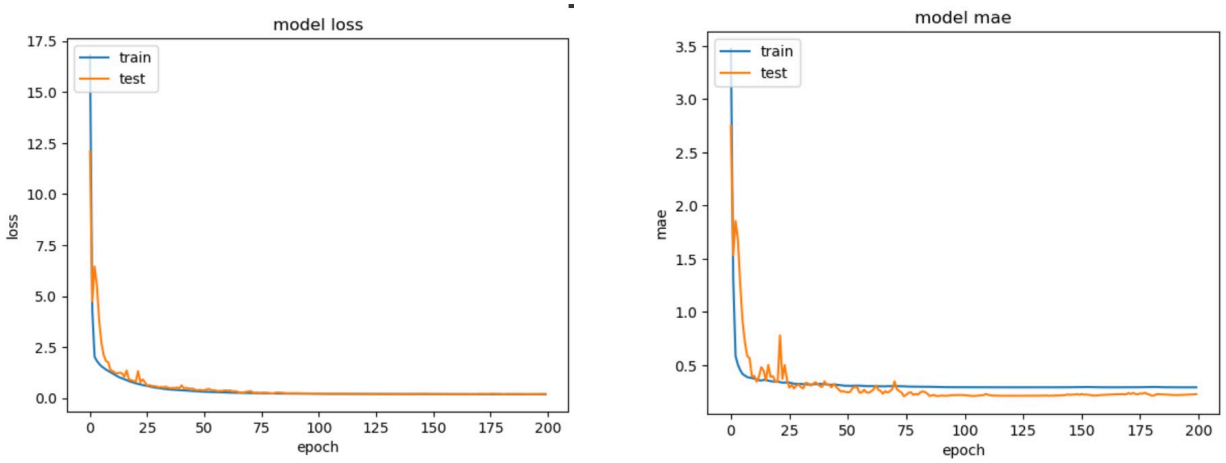
## Treinamento e avaliação

O modelo foi treinado por 200 épocas com um tamanho de lote original de 100 imagens, juntamente com 600 imagens de data augmentation, totalizando 700 imagens. As métricas de desempenho, incluindo o MAE, foram monitoradas ao longo do treinamento. O valor final de MAE foi de aproximadamente 7,32 pixels após a multiplicação para refletir a escala real.

O treinamento utilizou o método de `ReduceLROnPlateau`, que ajusta a taxa de aprendizado com base na performance do modelo, ajudando a evitar o overfitting.

## Resultados e conclusões

Pelo gráfico de **model Loss** e **model mae**, percebemos que não há overfitting significativo visto que os erros de validação estão próximos dos erros de treino, e isso se deve ao uso de data augmentation para aumentar a diversidade do conjunto de dados de treino, o que ajuda a evitar que o modelo fique muito ajustado aos dados específicos de treino. O uso de data augmentation introduz variações nos dados, permitindo que o modelo generalize melhor para novos dados de validação e teste, reduzindo o risco de overfitting.
Testamos se estavamos inserindo corretamente o conjunto de imagens para treino com data augmentation augmented_ax em ordem de equivalência com a coluna augmented_ay distorcida para considerar a translação das imagens. Assim, calculamos o MAE além de plotarmos as imagens de teste com as caixas de contornos do modelo de predição para localizar as bananas.
O tempo de processamento típico, usando a placa gráfica RTX 4060 do computador, leva cerca de 10 horas de compilação para prever o modelo, assim , buscamos salvar o modelo no arquivo 'modelo_trabalho_HAE_7.32.keras' e em 'inception_prelayer1.keras' para testes futuros sem precisar da necessidade de recompilar toda a rede neural.
O programa foi resolvido com resultados satisfatórios, demonstrando um erro de 7.32 pixels, um erro melhor do que o sugerido de 9 pixels.




**Figuras 1 e 2:** Gráficos do loss e do MAE para teste e treino durante cada época

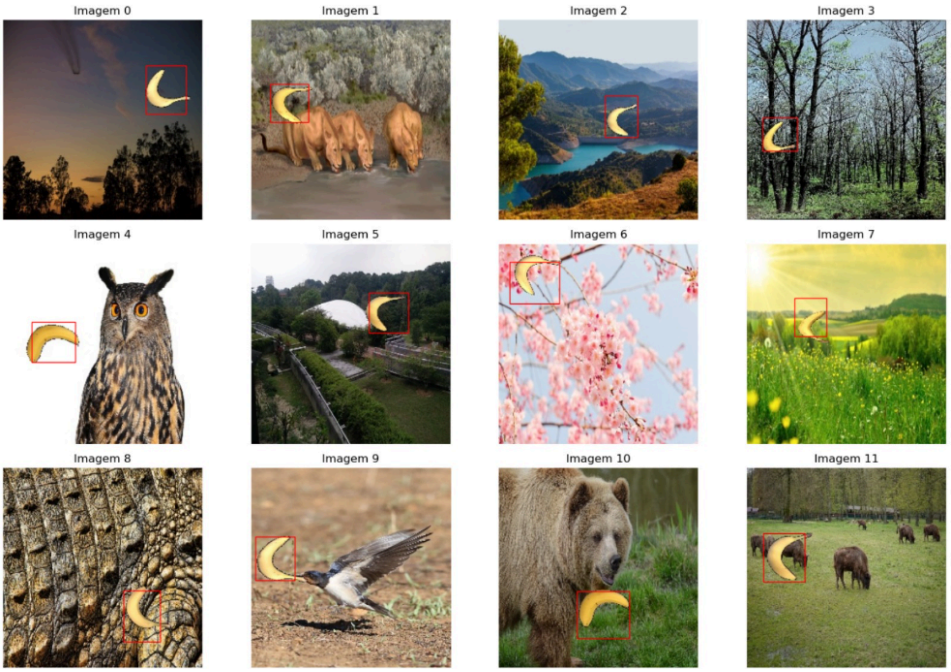



**Figura 3:** Resultados da detecção das bananas das 12 primeiras imagens

## REFERÊNCIAS:

Materiais presentes no site da disciplina:
Modelos pré-treinados para ImageNet; Cifar10; data augmentation; redes avançadas: VGG, GoogLeNet, ResNet.
http://www.lps.usp.br/hae/apostila/cifar-ead.pdf
get_config
https://scikit-learn.org/stable/modules/generated/sklearn.get_config.html
Object Detection: Resizing Bounding Box After Prediction
https://medium.com/@christopherhu1992/object-detection-resizing-bounding-box-after-prediction-fe44f03781a8