Atlântico Avanti

Demo day - Curso de Machine Learning

Equipe 2
Classificação de Alimentos: Food vs Non-Food

Atlântico Avanti

Equipe

Equipe 3

- Camila dos Santos Silva
- Eloane Laís Berto
- Felipe Barbosa de Lima
- Saimom Goz Siebem

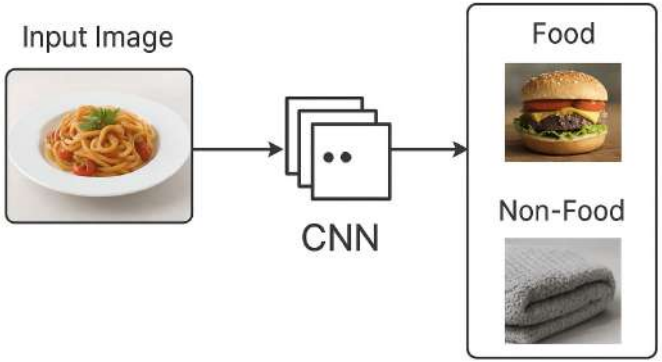
Sumário

- Introdução
- Metodologia
- Resultados
- Conclusões

Atlântico Avanti

Introdução

Introdução



- O projeto tem como objetivo classificar imagens em alimentos e não alimentos utilizando o Food-5K Image Dataset.

- Com foco é avaliar o desempenho das abordagens baseadas em Redes Neurais Convolucionais (CNNs).

Introdução

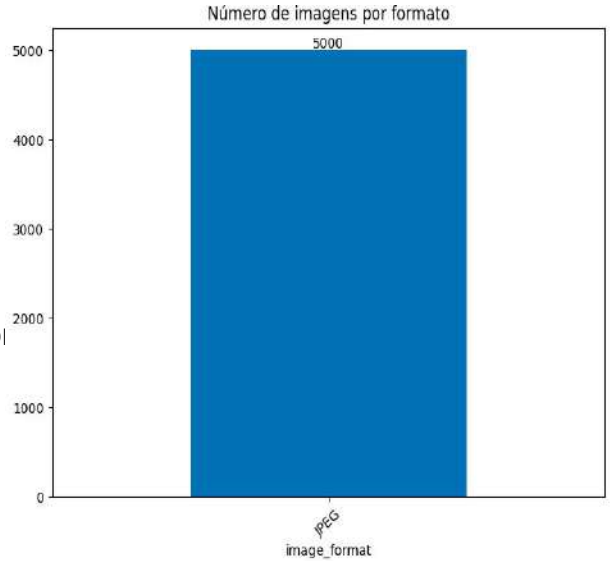
- Aplicações reais da classificação Food vs Non-Food:
  - Aplicativos de dieta e monitoramento alimentar
  - Recomendadores de cardápios e controle nutricional
  - Filtros de conteúdo alimentar em redes sociais

Atlântico Avanti

Metodologia

Metodologia

- Conjunto de 5000 imagens balanceadas em 2 classes:
  - food/ → imagens com comida
  - non_food/ → imagens sem comida
- Divisão:
  - Treinamento: 3.000
  - Validação: 1.000
  - Teste: 1.000 imagens
- Todas as imagens foram redimensio
  128 x 128 pixels
- Foram normalizadas entre 0 e 1



Metodologia

Quatro arquiteturas foram avaliadas para identificar a abordagem mais eficaz na classificação de imagens como **Food** ou **Non-Food**. Cada modelo representa uma estratégia diferente de design e otimização.

CNN Básica ->
Arquitetura da Literatura

CNN com Otimização ->
Ajuste fino de hiperparâmetros para melhor desempenho

CNN Leve -> Ajuste simplificada para eficiência

MobileNetV2 -> Transfer learning com modelo pré-treinado

Metodologia

Modelo 1 – CNN básica (modelo de referência).

- Modelo da Literatura

Camadas Principais:

- Conv2D com ativação ReLU
- MaxPooling2D para redução espacial
- Dropout para regularização
- Dense com ativação Sigmoid

Acurária Obtida: 83,6%

Metodologia

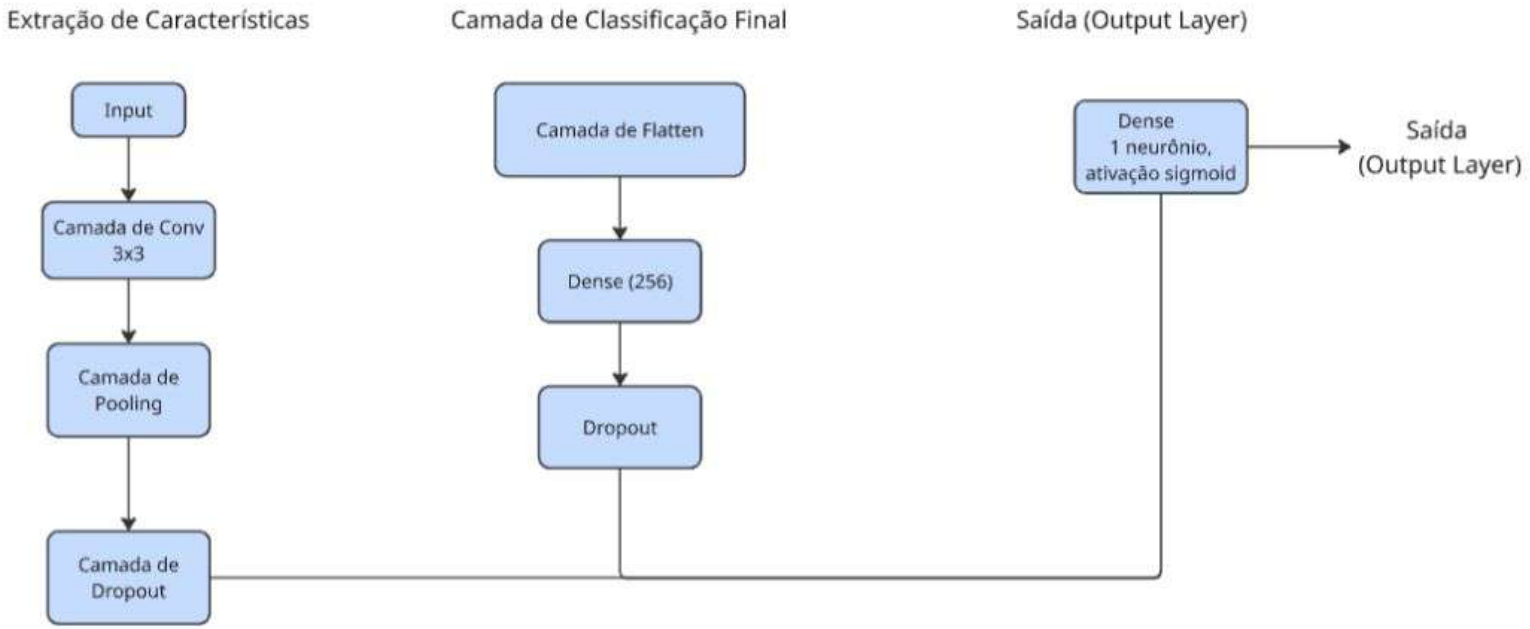
Modelo 2 – CNN com Otimização de Hiperparâmetros

- Estrutura mais leve, com redução do número de filtros.
- Aplicação de dropout mais moderado.
- Taxa de aprendizado reduzida (0.0005) para maior estabilidade -> optimizer = Adam(learning_rate=0.0005)

```
model_opt = tf.keras.Sequential()
# É onde o input é definido
# Camada cov
# Primeira convolução
model_opt.add(tf.keras.layers.Conv2D(32, kernel_size=(3,3), activation='relu', input_shape=(128,128,3)))
model_opt.add(tf.keras.layers.MaxPooling2D(2,2))
model_opt.add(tf.keras.layers.Dropout(0.25))

# Segunda convolução
model_opt.add(tf.keras.layers.Conv2D(64, kernel_size=(3,3), activation='relu'))
model_opt.add(tf.keras.layers.MaxPooling2D(2,2))
model_opt.add(tf.keras.layers.Dropout(0.25))

# Classificação Final
model_opt.add(tf.keras.layers.Flatten())
model_opt.add(tf.keras.layers.Dense(256, activation='relu'))

# Saída(Output Layer)
model_opt.add(tf.keras.layers.Dense(1, activation='sigmoid'))
```

Metodologia

Modelo 2 – CNN com Otimização de Hiperparâmetros



Metodologia

Modelo 3 – CNN Leve (Arquitetura Mais Leve)

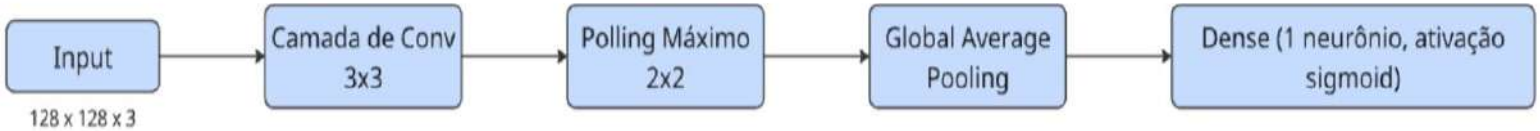
- Menos camadas convolucionais e uso de GlobalAveragePooling2D.
- Foco em eficiência e baixo custo computacional.
- Ideal para dispositivos com restrição de hardware.

Etapa	Função	Explicação resumida
Input	Entrada da imagem	Redimensionada e normalizada
Conv + Pooling	Extração de padrões visuais	Detecta bordas e texturas
GAP	Reduz parâmetros	Faz média por filtro → modelo leve
Dense (Sigmoid)	Classificação binária	Retorna probabilidade (0 = Non-Food, 1 = Food)

Metodologia

Modelo 3 – CNN Leve (Arquitetura Mais Leve)

Arquitetura Modelo:




```
model_light = tf.keras.Sequential([
    tf.keras.layers.Conv2D(32, (3,3), activation='relu', input_shape=(128,128,3)),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.GlobalAveragePooling2D(),
    # Camada densa de saída
    tf.keras.layers.Dense(1, activation='sigmoid')
])

model_light.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
```

Metodologia

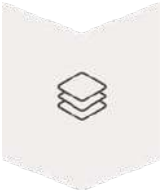
Modelo 4 – MobileNetV2

Este modelo aproveita o poder do transfer learning utilizando a arquitetura MobileNetV2 pré-treinada no dataset ImageNet. As camadas base congeladas preservam conhecimento visual genérico enquanto um classificador é treinado especificamente para a tarefa Food/Non-Food.

Metodologia

Modelo 4 – MobileNetV2

Base Pré-treinada - MobileNetV2 com pesos do ImageNet (camadas congeladas)



Classifier Head - GlobalAveragePooling2D → Dense(128, ReLU) → Dropout(0.3) → Sigmoid

Data Augmentation - Rotação, zoom, flip, shear para robustez

Metodologia

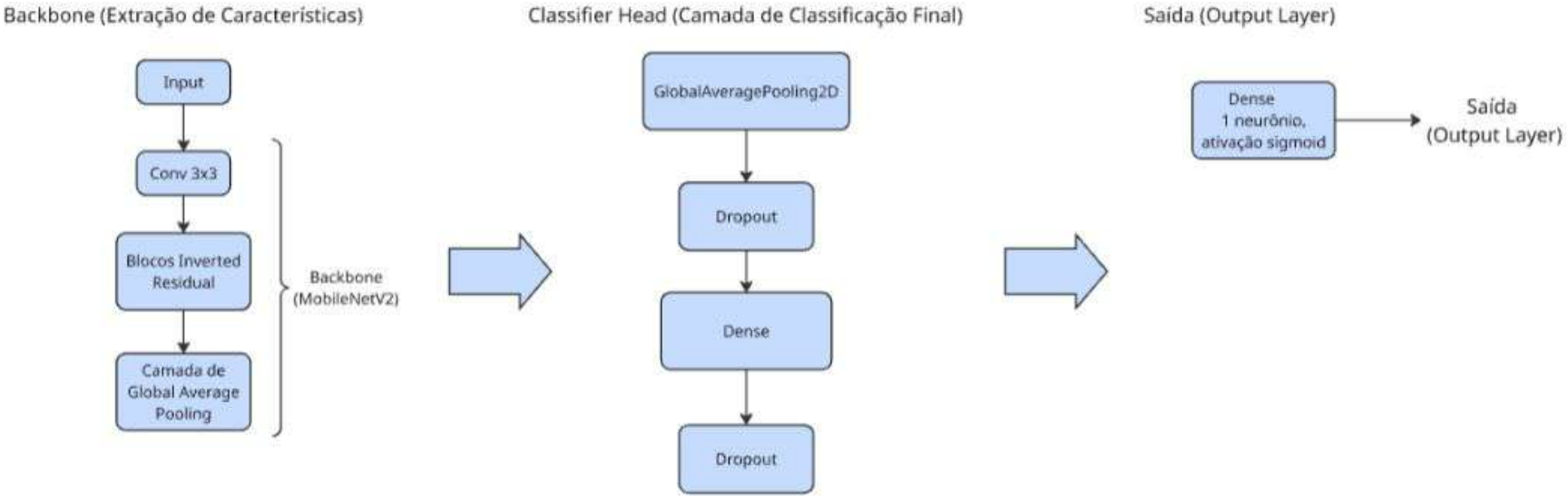
Modelo 4 – MobileNetV2

Etapa	Função	Explicação resumida
Input	Entrada da imagem	Recebe imagens 128x128x3 para análise
Backbone (MobileNetV2)	Extração de características	Rede pré-treinada no ImageNet — reconhece formas, texturas e padrões
GlobalAveragePooling2D	Redução de dimensionalidade	Resume cada mapa de ativação em um único valor médio
Dropout (0.3)	Regularização	Desativa 30% dos neurônios para evitar overfitting
Dense (128, ReLU)	Combinação de características	Interpreta e combina os padrões extraídos pela MobileNetV2
Dense (1, Sigmoid)	Classificação binária	Retorna probabilidade (0 = Non-Food, 1 = Food)

Metodologia

Modelo 4 – MobileNetV2

Arquitetura do modelo:

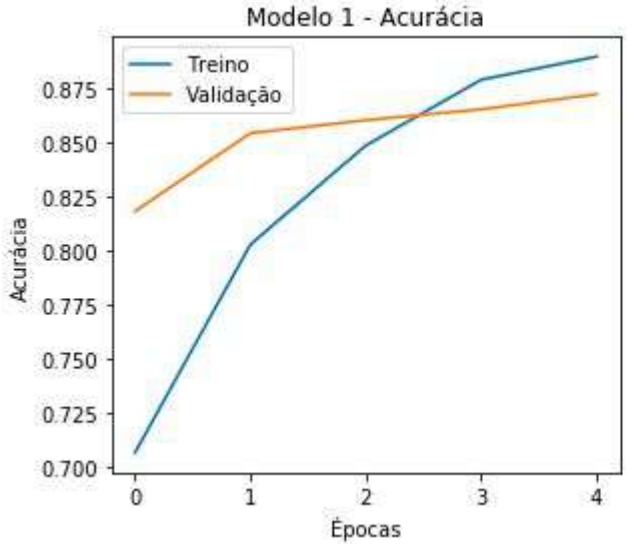


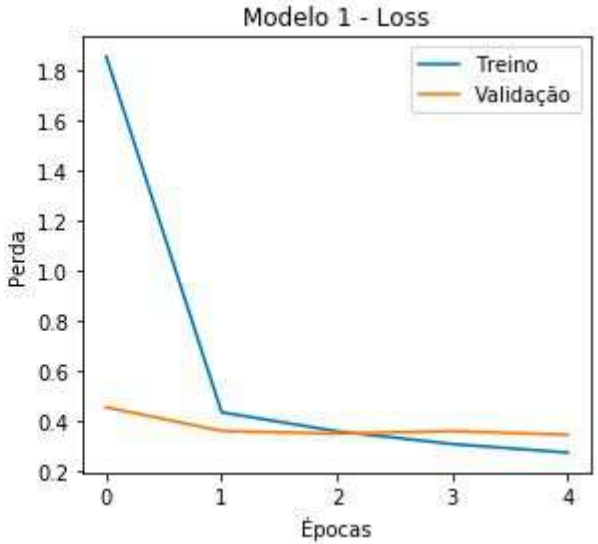
Atlântico Avanti

Resultados

Resultados

Modelo 1 – CNN básica (modelo de referência).

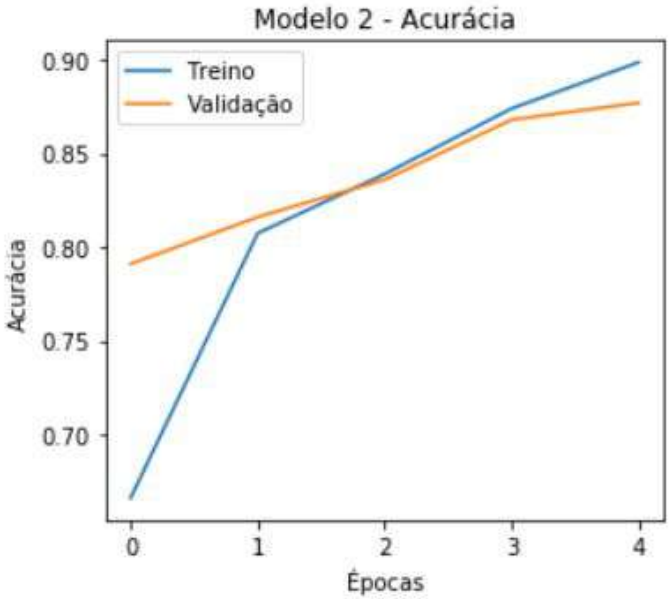


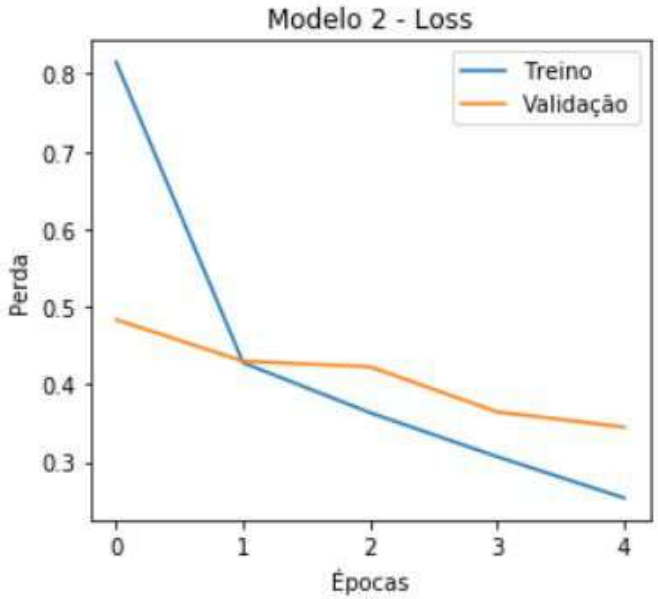


Acurária Obtida: 83,6%

Resultados

Modelo 2 – CNN com Otimização de Hiperparâmetros

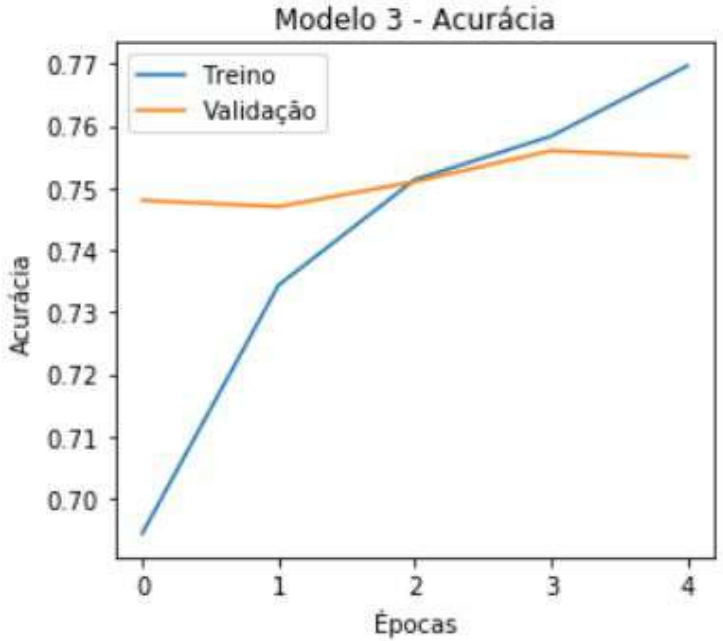


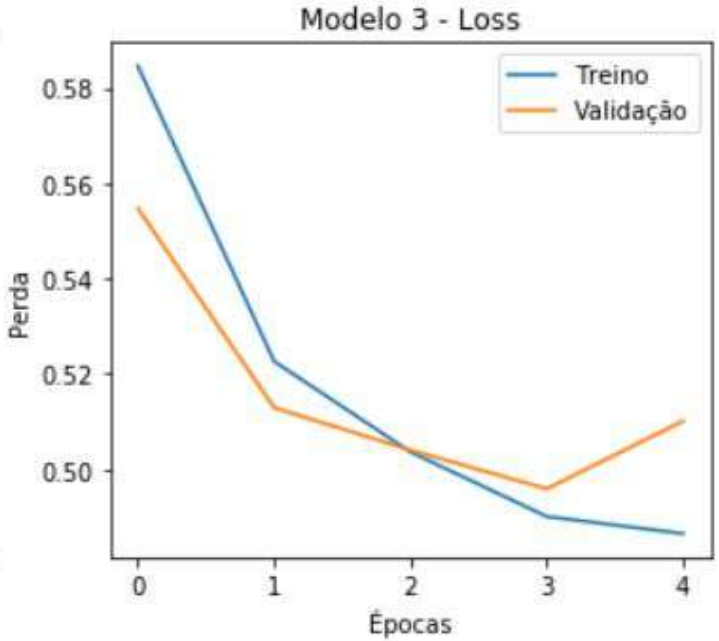


Acurácia: 85,6%

Resultados

Modelo 3 – CNN Leve (Arquitetura Mais Leve)

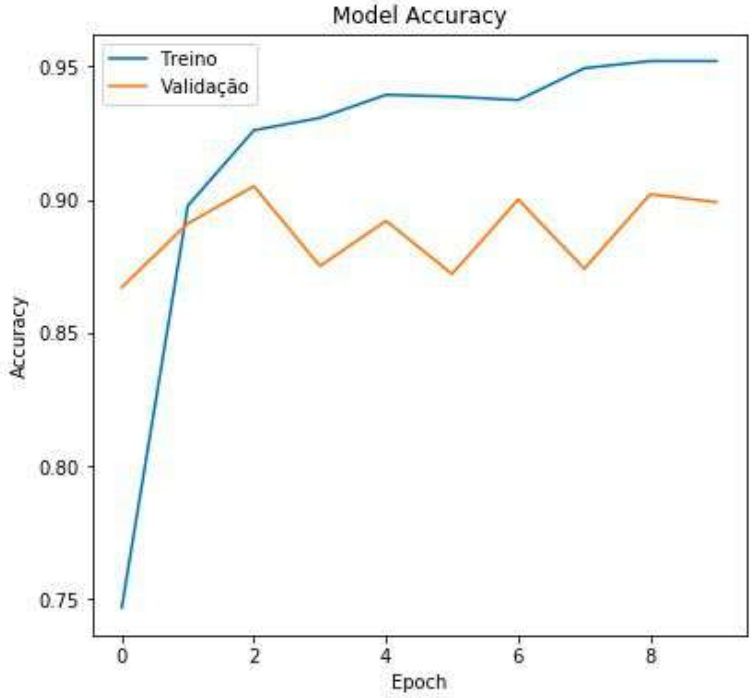


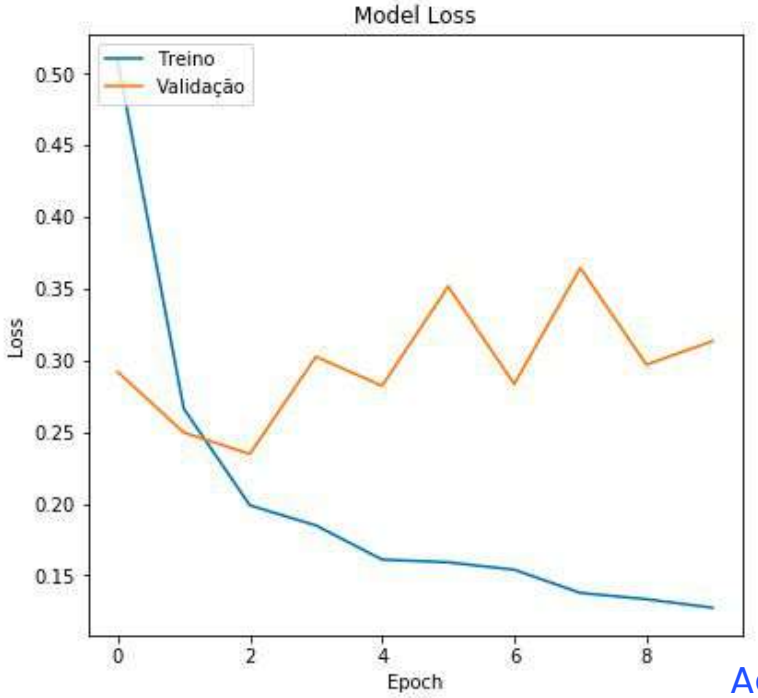


Acurácia: 75,7%

Resultados

Modelo 4 – MobileNetV2





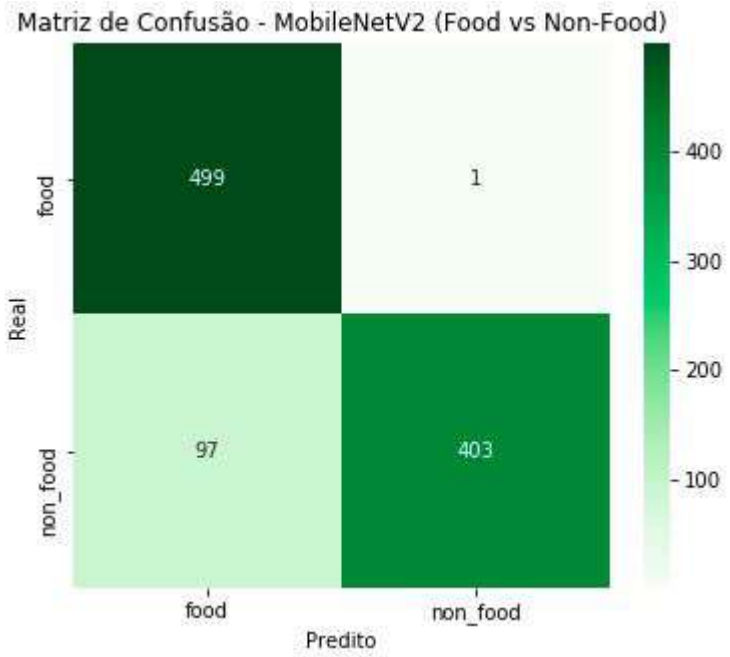
Acurácia: 90,2%

Resultados

Gráfico de desempenho: Modelo 4 - MobileNetV2

Classification Report:

	precision	recall	f1-score	support
food	0.84	1.00	0.91	500
non_food	1.00	0.81	0.89	500
accuracy			0.90	1000
macro avg	0.92	0.90	0.90	1000
weighted avg	0.92	0.90	0.90	1000



Resultados

Tabela. Matriz quadrada com linhas e colunas representando as classes.

	Previsto positivo	**Previsto negativo**
Positivo real	Verdadeiro positivo (VP)	Falso negativo (FN)
Negativo real	Falso positivo (FP)	Verdadeiro negativo (NV)

Componentes principais:

- Verdadeiro positivo (VP)
- Falso positivo (FP)
- Falso negativo (FN)
- Verdadeiro negativo (TN)

Resultados

Tabela de Falsos e Verdadeiros:

	Predito Positivo	Predito Negativo
Real Positivo	403	97
Real Negativo	1	499

Resultados

Acurácia

$$\text{Acurácia} = \frac{VP + VN}{VP + VN + FP + FN}$$

Substituindo:

$$\frac{403 + 499}{403 + 97 + 1 + 499} = \frac{902}{1000} = 0{,}902$$

90,2% de acerto

Precisão (classe positiva)

$$\text{Precisão} = \frac{VP}{VP + FP}$$

Substituindo:

$$\frac{403}{403 + 1} = 0{,}997$$

Excelente (quase sem falsos positivos)

Recall (sensibilidade)

$$\text{Recall} = \frac{VP}{VP + FN}$$

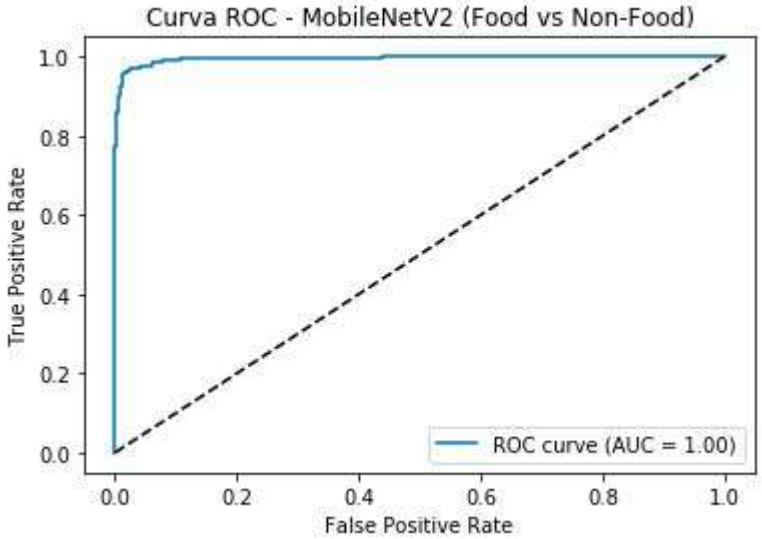
Substituindo:

$$\frac{403}{403 + 97} = 0{,}806$$

80,6%, indicando que o modelo **deixa escapar algumas imagens que são "comida"**.

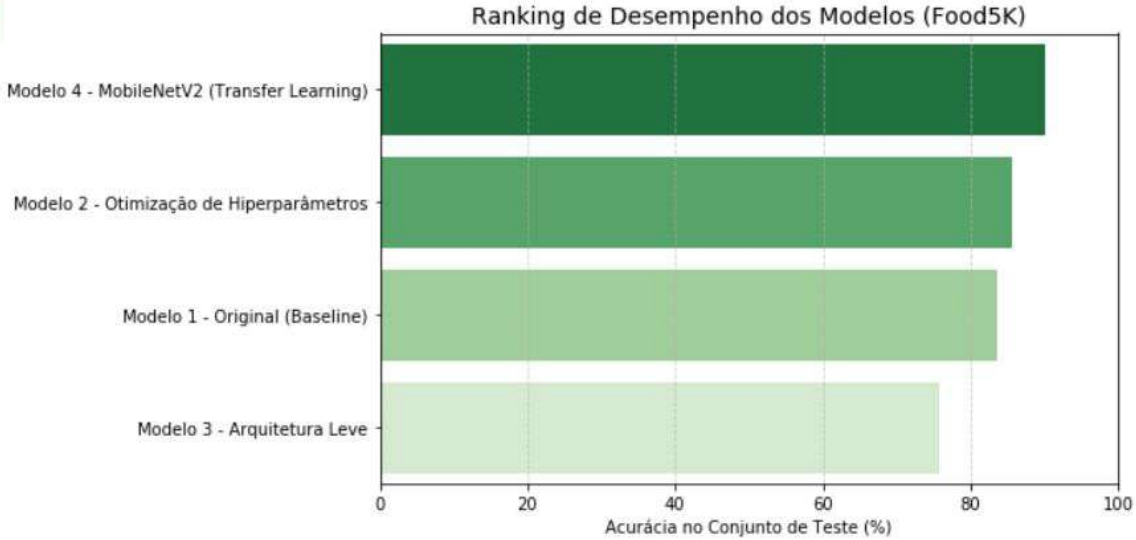
Resultados

Curva ROC e AUC



Resultados

	Modelo	Acurácia (%)
0	Modelo 4 - MobileNetV2 (Transfer Learning)	90.20
1	Modelo 2 - Otimização de Hiperparâmetros	85.60
2	Modelo 1 - Original (Baseline)	83.60
3	Modelo 3 - Arquitetura Leve	75.70



Atlântico Avanti

Conclusões

Conclusões

- Transfer Learning mostrou-se mais eficaz, especialmente com dataset limitado.
- Pesos pré-treinados permitiram extrair características mais robustas e generalizáveis.
- Ainda há espaço para melhoria (erros em imagens ambíguas).
- Desempenho pode ter sido influenciado por fatores como tamanho do dataset, balanceamento de classes e qualidade das imagens.

Conclusões

Sugestões para trabalhos futuros

- **Ampliar o dataset:** aumentar a variedade e quantidade de imagens para melhor generalização e menor overfitting.
- **Melhorar a qualidade das imagens:** aplicar pré-processamento (iluminação, contraste, remoção de ruído).
- **Balancear as classes:** usar oversampling ou undersampling para corrigir desequilíbrios entre food e non-food.

Conclusões

Sugestões para trabalhos futuros

- **Ajustar hiperparâmetros:** testar diferentes taxas de aprendizado, batch size, épocas e funções de ativação.

- **Explorar novas arquiteturas:** experimentar redes pré-treinadas como EfficientNet ou MobileNetV3 para melhor equilíbrio entre precisão e eficiência.

Referências Bibliográficas

FOOD-5K Dataset – Kaggle.

Howard et al. (2019). MobileNetV2: Inverted Residuals and Linear Bottlenecks.

Goodfellow, I., Bengio & Courville (2016). Deep Learning. MIT Press.

Chollet, F. (2017). Deep Learning with Python. Manning.