

Equipe - Classificação de produtos de comércio eletrônico

Alunos:

**Lucas Emanuel Rodrigues de Matos
Márcia Martins Leite de Sousa**

Sumário

- Introdução
- Metodologia
- Resultados
- Conclusões

Introdução

Resumo

do

problema:

Categorizar produtos de e-commerce em suas categorias corretamente.

Porque

esse

problema?

Ao atribuir corretamente os produtos em suas respectivas categorias melhoramos a experiência do usuário ao navegar pelo site melhorando a chance de aumentar as vendas.

Objetivo geral : Desenvolver um modelo eficiente de classificação de imagens de produtos usando técnicas de visão computacional.

Porque esse problema?

Objetivo geral

Metodologia

INFORMAÇÕES ESPERADAS

- Base de dados;
 - fonte, quantidade de imagens (por classes), mostrar uma imagem de cada classe;
- Método x - nome
 - ilustrar imagem original e processada (explicar cada imagem processada)

Dataset Notebooks


Metodologia

Base de dados

[ecommerce_product_images_18K
\(kaggle.com\)](https://www.kaggle.com/datasets/ecommerce-product-images-18k)

Filters

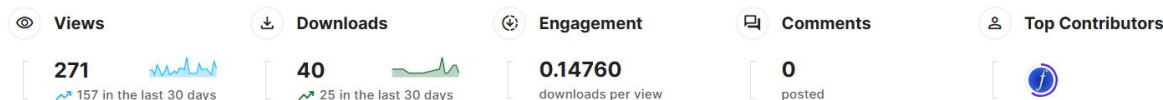
All Your Work Shared With You Bookmarks Hotness

**ecommerce_image_classification_mobilnetv2**

Updated 2mo ago
1 comment · ecommerce_product_images_18K

6

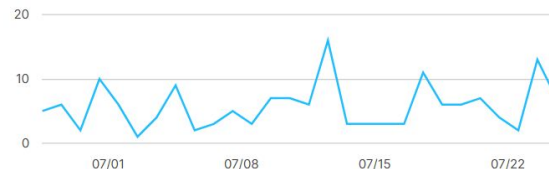
...



Detail View

Views

Last month



Downloads

Last month



Metodologia

○

Análise das imagens

As imagens desempenham um papel vital em vários domínios, incluindo visão computacional, aprendizado de máquina e análise de dados. **Compreender as propriedades e características** dos dados de imagem é **crucial** para extrair insights significativos e **tomar decisões** à partir do que foi **informado**.

Para a análise de imagens do Dataset, foram explorados os seguintes procedimentos:

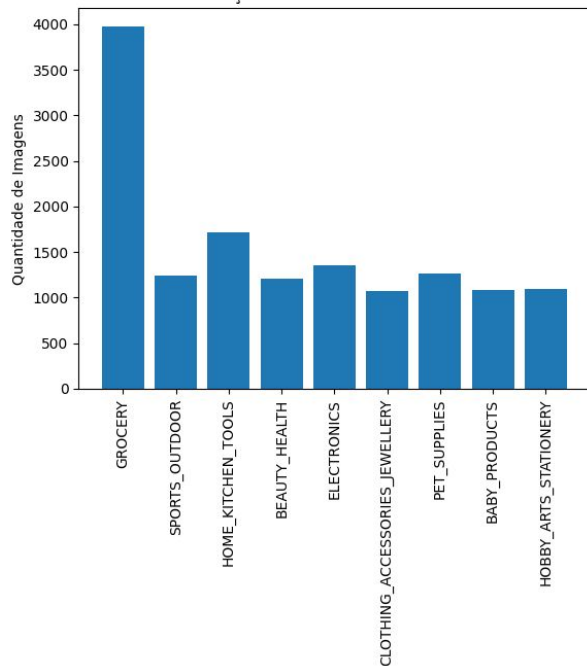
1. Integridade dos Arquivos
2. Consistência dos Metadados
3. Qualidade das Imagens
4. Distribuição das Classes
5. Duplicatas

[classificação-ecommerce.ipynb](#)
[- Colab](#)

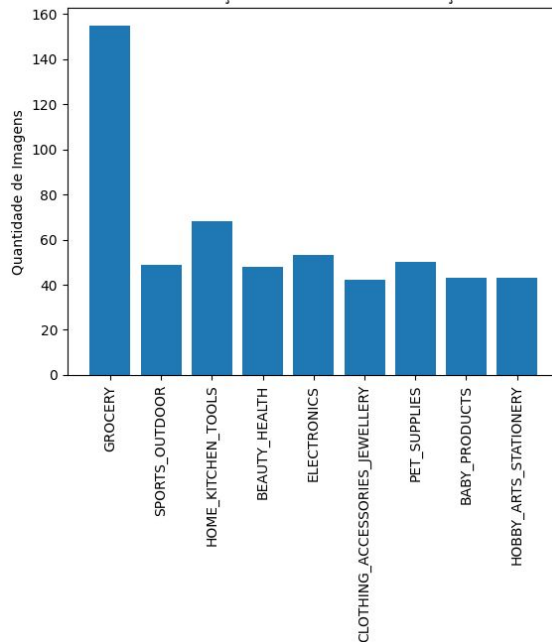
Metodologia

Distribuição das Classes

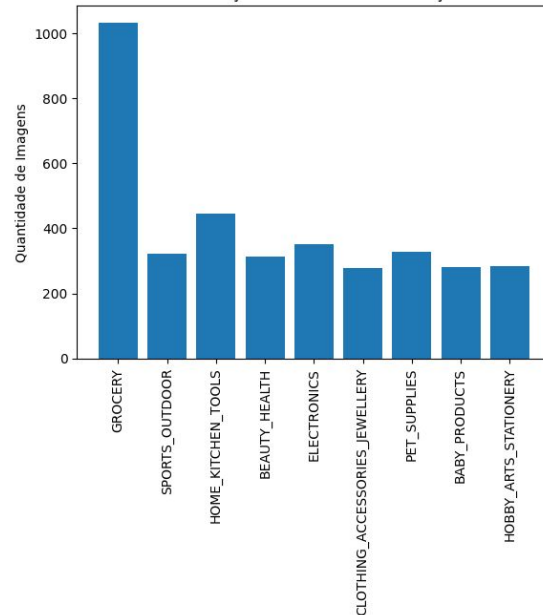
Distribuição das Classes de Treinamento



Distribuição das Classes de Verificação



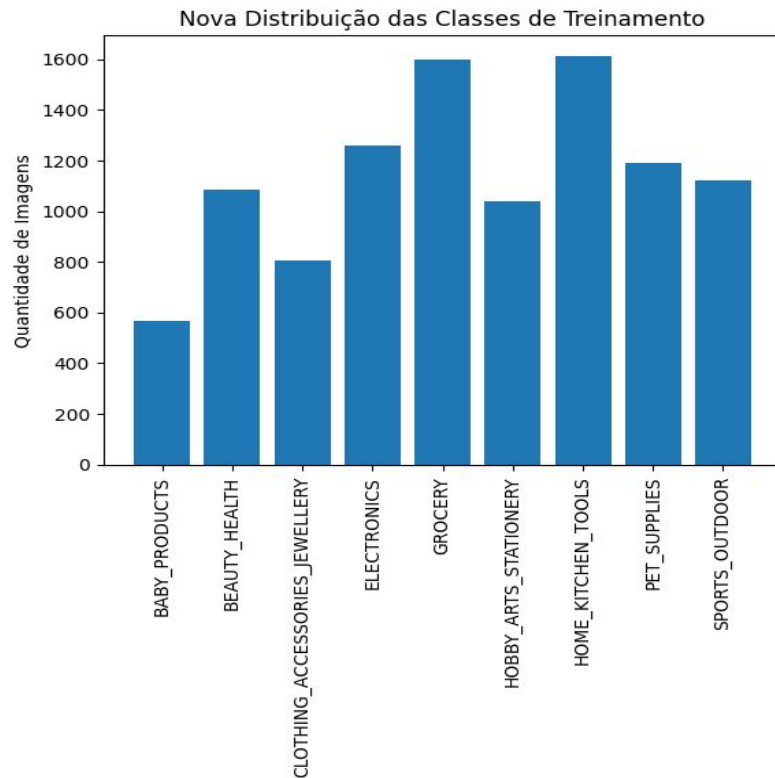
Distribuição das Classes de Validação



Metodologia

Balanceamento de classes

Reduzimos o GROCERY para ficar parelho com a classe com número maior de imagens



BABY_PROD
UCTS



BEAUTY_HEA
LTH



CLOTHING_ACCESS
ORIES_JEWELLERY



SPORTS_OUTDOOR



ELECTRONICS



HOME_KITCHEN_TOOLS



HOBBY_ARTS_STATIONERY



PET_SUPPLI



GROCERY



Metodologia

Pré-processamento

- **Normalização**
- **Redimensionar tamanho**
- **Conversão para escala de cinza**
- **Filtragem:**
 - **Detectação de bordas**
 - **Realce**
 - **Suavização**
 - **Correção de contraste**

Metodologia

Pré-processamento

- Normalização: escala /255

- Redimensionar tamanho

```
new_size = (300, 200)
```

- Conversão para escala de cinza

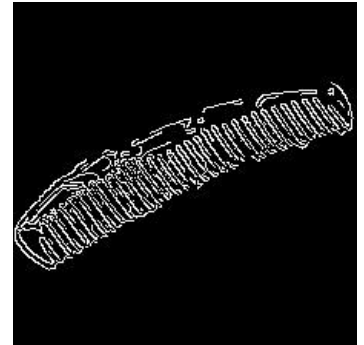


Metodologia

Pré-processamento

Filtragem:

- Detecção de bordas



- Realce



Metodologia

Pré-processamento

Filtragem:

- Suavização por mediana



- Correção de contraste



Metodologia

Aplicações de modelos

- MobileNetV2
- EfficientNet
- ResNet50
- CNN

Resultados

CNN

```
313/313 ————— 143s 420ms/step - accuracy: 0.1890 - loss: 2.2834 - val_accuracy: 0.2956 - val_loss: 2.0079
Epoch 2/10
 1/313 ————— 17s 55ms/step - accuracy: 0.2188 - loss: 2.0319/usr/lib/python3.10/contextlib.py:153: UserWarning:
    self.gen.throw(typ, value, traceback)
313/313 ————— 5s 15ms/step - accuracy: 0.2188 - loss: 2.0319 - val_accuracy: 0.1250 - val_loss: 2.0668
Epoch 3/10
313/313 ————— 121s 378ms/step - accuracy: 0.2581 - loss: 2.0019 - val_accuracy: 0.3769 - val_loss: 1.7714
Epoch 4/10
313/313 ————— 5s 16ms/step - accuracy: 0.3750 - loss: 1.7858 - val_accuracy: 0.0000e+00 - val_loss: 2.733:
Epoch 5/10
313/313 ————— 136s 373ms/step - accuracy: 0.2915 - loss: 1.9208 - val_accuracy: 0.4096 - val_loss: 1.6717
Epoch 6/10
313/313 ————— 1s 2ms/step - accuracy: 0.4062 - loss: 1.8254 - val_accuracy: 0.0625 - val_loss: 2.6768
Epoch 7/10
313/313 ————— 144s 382ms/step - accuracy: 0.3071 - loss: 1.8619 - val_accuracy: 0.4234 - val_loss: 1.6332
Epoch 8/10
313/313 ————— 0s 85us/step - accuracy: 0.3125 - loss: 1.9583 - val_accuracy: 0.2500 - val_loss: 2.6927
Epoch 9/10
313/313 ————— 120s 375ms/step - accuracy: 0.3384 - loss: 1.8355 - val_accuracy: 0.4220 - val_loss: 1.6295
Epoch 10/10
313/313 ————— 0s 740us/step - accuracy: 0.2188 - loss: 2.1884 - val_accuracy: 0.3750 - val_loss: 2.3594
```

Resultados

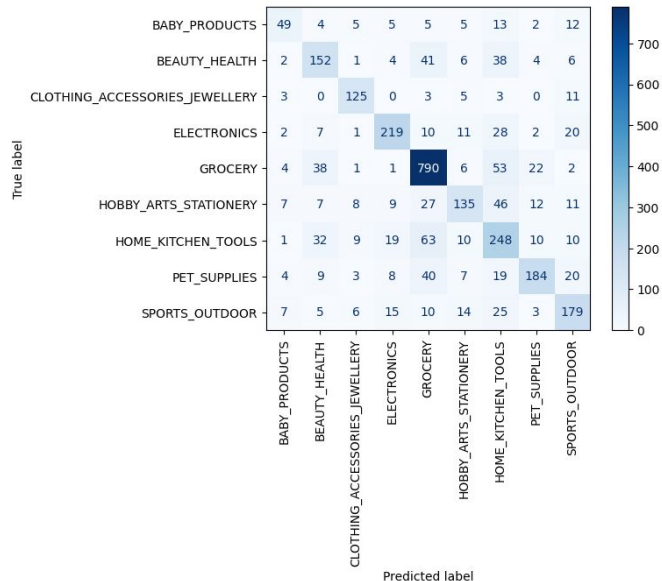
EfficientNet

1 Tentativa:

```
Epoch 1/30
322/322 [=====] - 107s 332ms/step - loss: 3.2137 - accuracy: 0.1067 - val_loss: 3.1054 - val_accuracy: 0.1026 - lr: 1.0000e-04
Epoch 2/30
322/322 [=====] - 108s 334ms/step - loss: 3.0031 - accuracy: 0.1231 - val_loss: 2.0778 - val_accuracy: 0.3116 - lr: 1.0000e-04
Epoch 3/30
322/322 [=====] - 108s 336ms/step - loss: 2.9246 - accuracy: 0.1192 - val_loss: 2.1525 - val_accuracy: 0.2681 - lr: 1.0000e-04
Epoch 4/30
322/322 [=====] - 104s 321ms/step - loss: 2.8601 - accuracy: 0.1220 - val_loss: 2.1109 - val_accuracy: 0.2491 - lr: 1.0000e-04
Epoch 5/30
322/322 [=====] - 96s 298ms/step - loss: 2.7762 - accuracy: 0.1266 - val_loss: 2.2176 - val_accuracy: 0.0870 - lr: 1.0000e-04
Epoch 6/30
322/322 [=====] - 117s 362ms/step - loss: 2.7596 - accuracy: 0.1224 - val_loss: 2.1268 - val_accuracy: 0.1886 - lr: 1.0000e-05
Epoch 7/30
322/322 [=====] - 103s 320ms/step - loss: 2.7545 - accuracy: 0.1264 - val_loss: 2.1523 - val_accuracy: 0.1424 - lr: 1.0000e-05
Training complete.
```

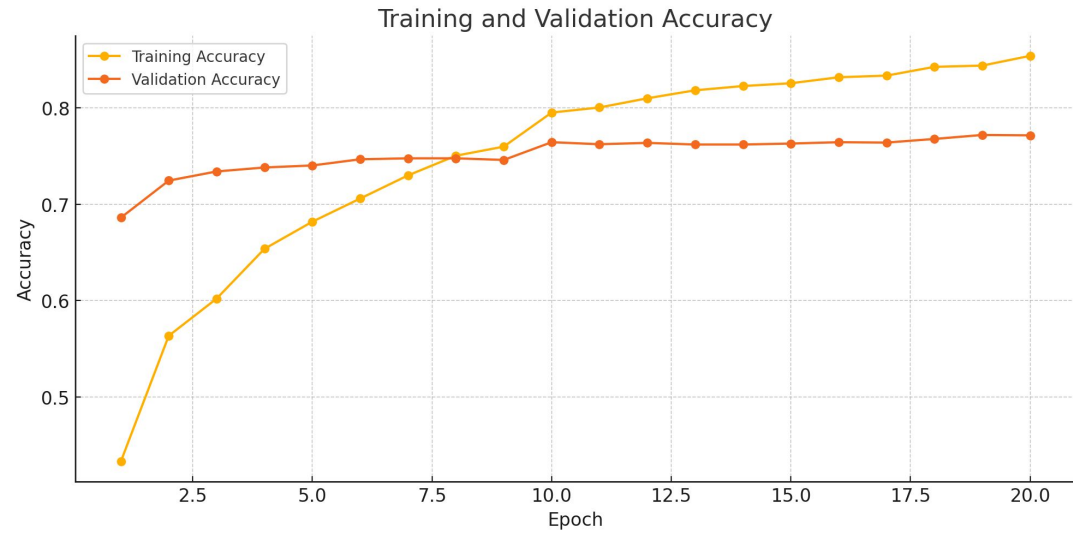
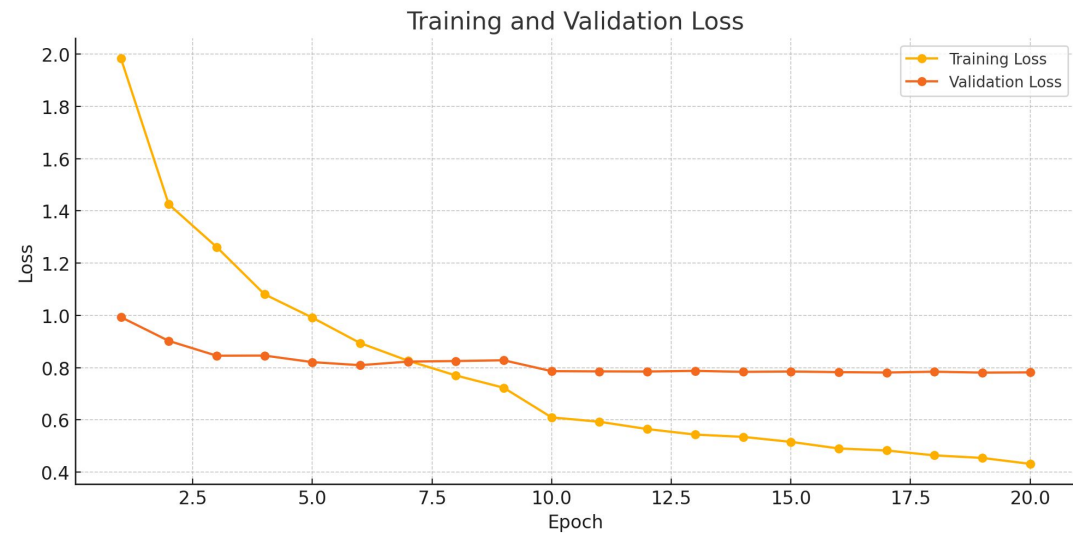
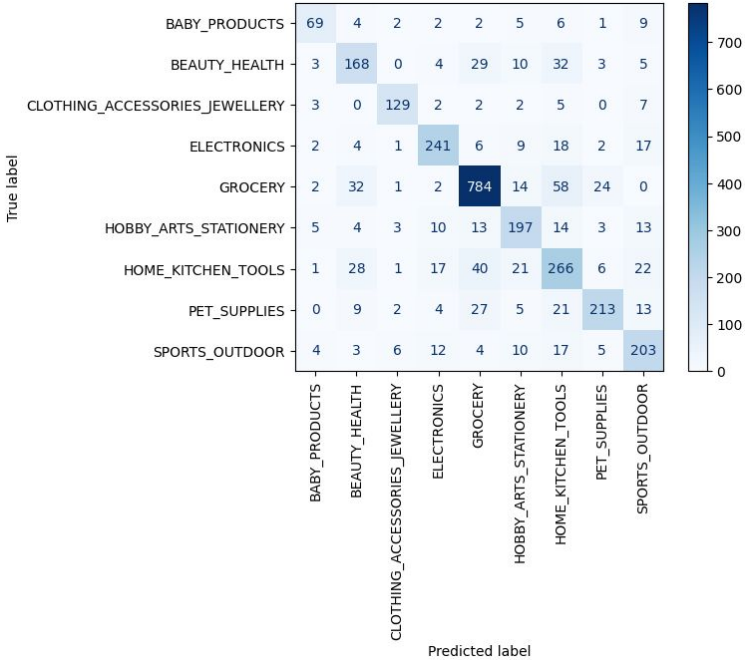

Resultados

EfficientNet



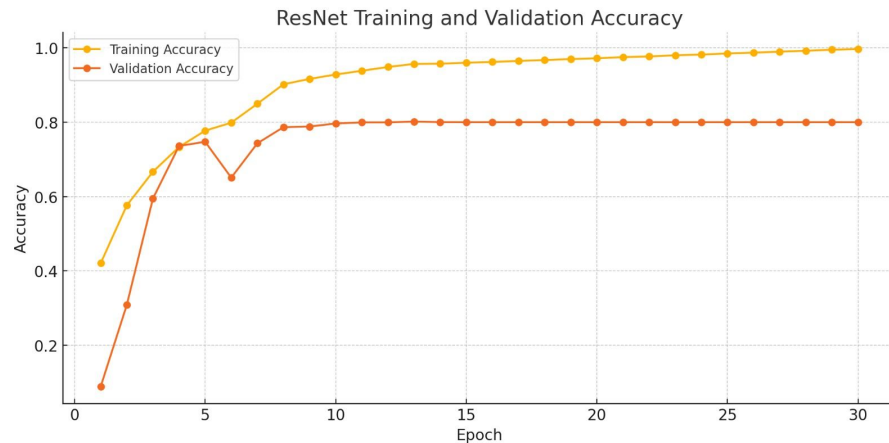
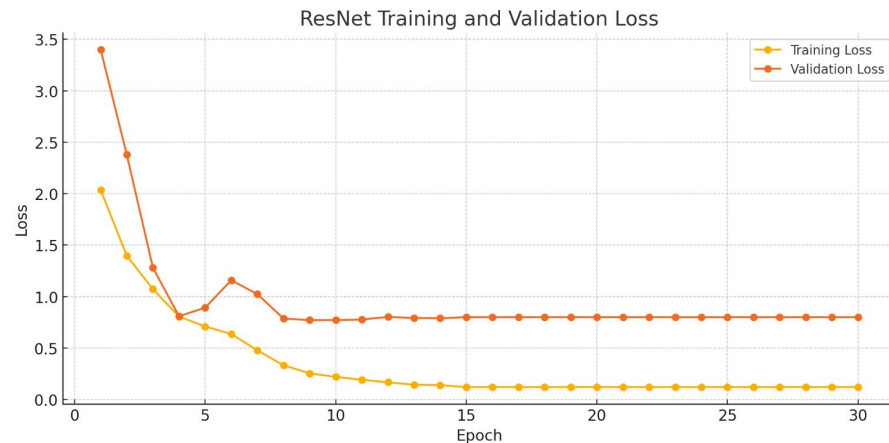
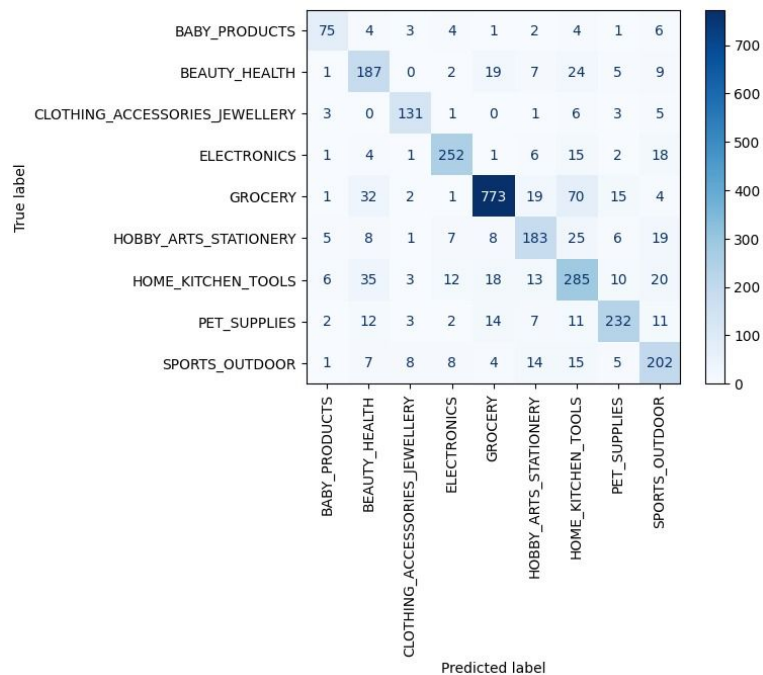
Resultados

MobileNet



Resultados

ResNet50



Resultado - Observação

Chosen Image Actual Category: GROCERY



Image 1 - Actual Category: GROCERY

Model Predicts:

	Class	Probability
0	GROCERY	0.94
1	HOME_KITCHEN_TOOLS	0.06
2	PET_SUPPLIES	0.00
3	CLOTHING_ACCESSORIES_JEWELLERY	0.00
4	HOBBY_ARTS_STATIONERY	0.00
5	BEAUTY_HEALTH	0.00
6	ELECTRONICS	0.00
7	BABY_PRODUCTS	0.00
8	SPORTS_OUTDOOR	0.00

1/1 [=====] - 0s 30ms/step

Chosen Image Actual Category: BEAUTY_HEALTH

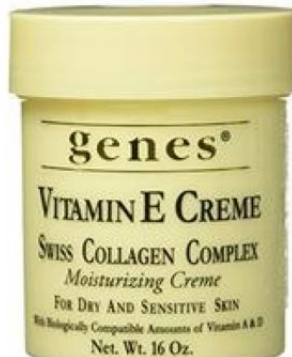


Image 4 - Actual Category: BEAUTY_HEALTH

Model Predicts:

	Class	Probability
0	BEAUTY_HEALTH	0.99
1	GROCERY	0.01
2	PET_SUPPLIES	0.00
3	SPORTS_OUTDOOR	0.00
4	CLOTHING_ACCESSORIES_JEWELLERY	0.00
5	HOME_KITCHEN_TOOLS	0.00
6	HOBBY_ARTS_STATIONERY	0.00
7	BABY_PRODUCTS	0.00
8	ELECTRONICS	0.00

1/1 [=====] - 0s 27ms/step

Chosen Image Actual Category: SPORTS_OUTDOOR



Image 6 - Actual Category: SPORTS_OUTDOOR

Model Predicts:

	Class	Probability
0	SPORTS_OUTDOOR	1.0
1	BEAUTY_HEALTH	0.0
2	PET_SUPPLIES	0.0
3	HOME_KITCHEN_TOOLS	0.0
4	ELECTRONICS	0.0
5	HOBBY_ARTS_STATIONERY	0.0
6	BABY_PRODUCTS	0.0
7	GROCERY	0.0
8	CLOTHING_ACCESSORIES_JEWELLERY	0.0

1/1 [=====] - 0s 28ms/step

Resultado - Observação

Chosen Image Actual Category: CLOTHING_ACCESSORIES_JEWELLERY



Image 3 - Actual Category: CLOTHING_ACCESSORIES_JEWELLERY

Model Predicts:

	Class	Probability
0	CLOTHING_ACCESSORIES_JEWELLERY	1.0
1	BABY_PRODUCTS	0.0
2	HOBBY_ARTS_STATIONERY	0.0
3	GROCERY	0.0
4	HOME_KITCHEN_TOOLS	0.0
5	SPORTS_OUTDOOR	0.0
6	BEAUTY_HEALTH	0.0
7	PET_SUPPLIES	0.0
8	ELECTRONICS	0.0

1/1 [=====] - 0s 30ms/step

Chosen Image Actual Category: BABY_PRODUCTS



Image 2 - Actual Category: BABY_PRODUCTS

Model Predicts:

	Class	Probability
0	CLOTHING_ACCESSORIES_JEWELLERY	0.64
1	BABY_PRODUCTS	0.31
2	SPORTS_OUTDOOR	0.02
3	PET_SUPPLIES	0.01
4	HOBBY_ARTS_STATIONERY	0.01
5	GROCERY	0.00
6	HOME_KITCHEN_TOOLS	0.00
7	ELECTRONICS	0.00
8	BEAUTY_HEALTH	0.00

1/1 [=====] - 0s 31ms/step

Resultado – Observação

```
base_model = ResNet50(weights='imagenet', include_top=False, input_shape=(224, 224, 3))

# Add out top layers
x = base_model.output
x = Flatten()(x)
x = Dense(512, activation='relu')(x) # Ajuste conforme necessário
x = BatchNormalization()(x)
x = Dropout(0.5)(x)
x = Dense(256, activation='relu')(x)
x = BatchNormalization()(x)
x = Dropout(0.5)(x)

predictions = Dense(num_classes, activation='softmax')(x)

# Combine the base layer and our top layers
model = Model(inputs=base_model.input, outputs=predictions)

# Freeze the base model layers
for layer in base_model.layers[-20:]:
    layer.trainable = True

# Choose the optimizer from optimizers dictionary
optimizer_ = optimizers['Adam']

# Compile the model
model.compile(optimizer=optimizer_, loss='categorical_crossentropy', metrics=['accuracy'])

print("Model compiled and ready for training.")
```

Conclusão

Modelo	Acurácia	Épocas
CNN	0.2188	10
EfficientNet B0	0.1264	30
MobileNetv2	0.8734	20
EfficientNet B0	0.6343	30
ResNet50	0.9716	30

Conclusões

O estudo contribuiu significativamente para o desenvolvimento de uma base prática sólida para estudos futuros. Além disso, os testes realizados permitiram identificar um método de aplicação de filtros que **supera o existente no Kaggle**, proporcionando **melhorias no pré-processamento**.

Como próximo passo, pretendemos subir o notebook aprimorado para o Kaggle, de modo a compartilhar esses avanços com a comunidade e fomentar novas pesquisas que sigam a mesma linha de aprimoramento.

Trabalhos Futuros:

- Testes com mais variabilidade de filtros
- Teste com mais variações de camadas de ativação
- Aplicação de modelos que seguem o princípio Transfer Learning