# AI & SUSTAINABILITY IN VET EDUCATION

**Nome do(s) autor(es):** Sucessos Criativos, Lda

**Título do WP:** WP3 – IA-SAVES@ProjectHub

**Referência de Atividade:** 3.2 – Modelo de projeto para IA-SAVES@ProjectHub

Co-funded by the European Union

Este projeto foi financiado com o apoio da Comissão Europeia. Esta comunicação reflete apenas as opiniões do autor, e a Comissão não pode ser responsabilizada por qualquer uso que possa ser feito da informação nela contida.

# Índice

1. Título do Projeto: 4
2. Categoria do Projeto: 4
3. Descrição do Projeto: 4
4. Palavras-chave: 4
5. Público-alvo: 4
6. Objetivos do Projeto: 4
7. Cronologia do Projeto: 5
8. Recursos Fornecidos: 5
9. Ferramentas e Tecnologias Utilizadas: 6
10. Plano de Implementação Passo a Passo: 7
**Passo 1: Prepare o Raspberry Pi para Interação de Hardware 7**
**Passo 2: Ligue o Sensor de LDR 7**
**Passo 3: Escreve o programa Python para Detetar Luz 8**
11. Critérios de Avaliação: 12
12. Línguas Disponíveis: 12
13. Colaboradores: 12
17. Feedback e Melhoria Contínua: 13

Co-funded by the European Union

Este projeto foi financiado com o apoio da Comissão Europeia. Esta comunicação reflete apenas as opiniões do autor, e a Comissão não pode ser responsabilizada por qualquer uso que possa ser feito da informação nela contida.

**1. Título do Projeto:**

Classificação de Imagens com MobileNet e Aprendizagem por Transferência

---

**2. Categoria do Projeto:**

IA na Educação

---

**3. Descrição do Projeto:**

Neste microprojeto, os alunos irão desenvolver um modelo personalizado de classificação de imagens capaz de distinguir entre três tipos de flores: **rosas**, **girassóis** e **tulipas**. O projeto utiliza **o MobileNetV3**, um modelo de aprendizagem profunda leve bem adequado para dispositivos de borda, e utiliza **aprendizagem por transferência** para afinar o modelo usando um pequeno conjunto de dados curado.

Este projeto é ideal para aprendizes interessados em visão computacional, processamento de imagem e treino de modelos utilizando arquiteturas pré-treinadas. Demonstra as melhores práticas para preparar conjuntos de dados, lidar com o desequilíbrio de classes e estruturar dados para treino com TensorFlow e Keras.

---

**4. Palavras-chave:**

Transfer Learning, MobileNetV3, Classificação de Imagens, Conjunto de Dados Desequilibrados, ImageDataGenerator, TensorFlow / Keras

---

Co-funded by the European Union

Este projeto foi financiado com o apoio da Comissão Europeia. Esta comunicação reflete apenas as opiniões do autor, e a Comissão não pode ser responsabilizada por qualquer uso que possa ser feito da informação nela contida.

**5. Público-alvo:**

Educadores, aprendizes e criadores, com ou sem experiência prévia, que estão a iniciar a sua jornada em IA.

---

**6. Objetivos do Projeto:**

Após concluírem este projeto, os alunos poderão:

- Compreender como organizar e pré-processar conjuntos de dados para classificação de imagens.
- Lidar com conjuntos de dados desequilibrados usando técnicas de aumento de dados.
- Use o ImageDataGenerator do TensorFlow para carregar, aumentar e agrupar imagens.
- Aplica **a aprendizagem por transferência** usando o MobileNetV3 para construir um classificador personalizado.
- Treine, valide e teste um modelo de deep learning em conjuntos de dados personalizados de flores.
- Avalie o desempenho do modelo usando matrizes de precisão, precisão, recordação e confusão.
- Faça previsões sobre novas imagens e interprete os resultados dos modelos.

---

**7. Cronologia do Projeto:**

A duração estimada é de aproximadamente **2,5 a 3 horas**.

---

Co-funded by the European Union

Este projeto foi financiado com o apoio da Comissão Europeia. Esta comunicação reflete apenas as opiniões do autor, e a Comissão não pode ser responsabilizada por qualquer uso que possa ser feito da informação nela contida.

**8. Recursos Fornecidos:**

- Para estas tarefas pode usar tanto um computador como o seu raspberry pi.

---

**9. Ferramentas e Tecnologias Utilizadas:**

**Google Colab**: Ambiente de desenvolvimento baseado na cloud

**TensorFlow** / Keras: Estrutura de aprendizagem profunda

**Matplotlib, NumPy**: Visualização e operações numéricas

**Kaggle**: Fonte do conjunto de dados de imagens de flores rotuladas

---

**10. Plano de Implementação Passo a Passo:**

**ANTES DE COMEÇARES**

Este projeto irá guiá-lo no treino de um modelo de classificação de imagens para reconhecer três tipos de flores: **rosas**, **girassóis** e **tulipas**. Vais usar **o Google Colab**, **TensorFlow** e **MobileNetV3 com aprendizagem por transferência**.

Requisitos

- Uma Conta Google
- Ligação à Internet
- Acesso a Kaggle
- Pesquisa Colab no Google (não é necessária instalação)

Configuração de Pastas

Co-funded by the European Union

Este projeto foi financiado com o apoio da Comissão Europeia. Esta comunicação reflete apenas as opiniões do autor, e a Comissão não pode ser responsabilizada por qualquer uso que possa ser feito da informação nela contida.

1. Vá ao Kaggle Flowers Dataset, descarregue o conjunto de dados e descompacte-o.

2. Mantenha apenas as seguintes pastas do conjunto de dados:

    - `Rose/`
    - `Girassol/`
    - `tulipa/`

3. No seu **Google Drive**, crie a estrutura de pastas:

```
/O Meu Impulso/
└── Treino/
    ├── datasets/
    │   └── flores/
    │   ├── Rose/
    │   ├── girassol/
    │   └── tulipa/
    ├── modelos/
    └── ImagensSuporte/
```

4. Carrega as tuas três pastas de flores para `/datasets/flowers/`.

5. Cria uma pasta `/models/` vazia (para guardar o modelo).

6. Abra este caderno no Colab e guarde-o em `/Training/`.

7. Ativar Índice em Colab:
   **Ver > Índice**

Co-funded by the European Union

Este projeto foi financiado com o apoio da Comissão Europeia. Esta comunicação reflete apenas as opiniões do autor, e a Comissão não pode ser responsabilizada por qualquer uso que possa ser feito da informação nela contida.

**Passo 1: MONTAR O GOOGLE DRIVE**

Monte o seu Google Drive no Colab para poder ler e guardar ficheiros:

```
Do Google.colab Import Drive

drive.mount('/content/drive')
```

---

**Passo 2: RECOLHA DE DADOS**

Estamos a construir um modelo para classificar **rosas**, **girassóis** e **tulipas**. As imagens estão organizadas em pastas assim:

```
/Treino/conjuntos de dados/flores/
  ├── Rose/
  ├── girassol/
  └── tulipa/
```

Cada pasta deve conter imagens apenas para essa classe.

---

**Passo 3: VERIFICAÇÃO E TRATAMENTO DE DADOS DESEQUILIBRADOS**

O passo seguinte é verificar dados balanceados como parte da fase de preparação e pré-processamento dos dados. Esta verificação deve idealmente ocorrer após

Co-funded by the European Union

Este projeto foi financiado com o apoio da Comissão Europeia. Esta comunicação reflete apenas as opiniões do autor, e a Comissão não pode ser responsabilizada por qualquer uso que possa ser feito da informação nela contida.

carregar e organizar o conjunto de dados, mas antes de começar o treino, o que significa que este é o momento para a fazer.

Verificar o equilíbrio dos dados é essencial porque um conjunto de dados desequilibrado — onde algumas classes têm significativamente mais imagens do que outras — pode levar a um modelo tendencioso que favorece as classes sobre-representadas.

Um conjunto de dados equilibrado garante que o modelo tem oportunidades iguais para aprender as características de cada classe, melhorando a sua capacidade de classificar cada categoria com precisão.

Se verificarmos que o conjunto de dados está desequilibrado, podemos tomar ações corretivas, como aumentar imagens em turmas sub-representadas ou aplicar pesos de classe durante o treino, para resolver este problema antes que o modelo comece a aprender.

```
Importação de sistemas operativos

de coleções importação Contador

importar matplotlib.pyplot como plt



# Caminho para o diretório principal do conjunto de dados

data_dir                    =                    '/content/drive/My
Drive/TrainingGreeceCV/datasets/flowers/'



# Contar imagens em cada subdiretório (ou seja, cada classe)

class_counts = Contador()

Para class_name em os.listdir(data_dir):
```

Co-funded by the European Union

Este projeto foi financiado com o apoio da Comissão Europeia. Esta comunicação reflete apenas as opiniões do autor, e a Comissão não pode ser responsabilizada por qualquer uso que possa ser feito da informação nela contida.

```
    class_path = os.path.join(data_dir, class_name)

    if os.path.isdir(class_path):

 class_counts[class_name] = len(os.listdir(class_path))

# Imprimir o número de imagens em cada aula

Para class_name, conte em class_counts.itens():

    print(f"Class '{class_name}': {count} images")



# Traçar a distribuição de imagens por classe

plt.figure(figsize=(8, 6))

plt.bar(class_counts.keys(),   class_counts.values(),   color='azul-
celeste')

plt.xlabel('Classe')

plt.ylabel('Número de Imagens')

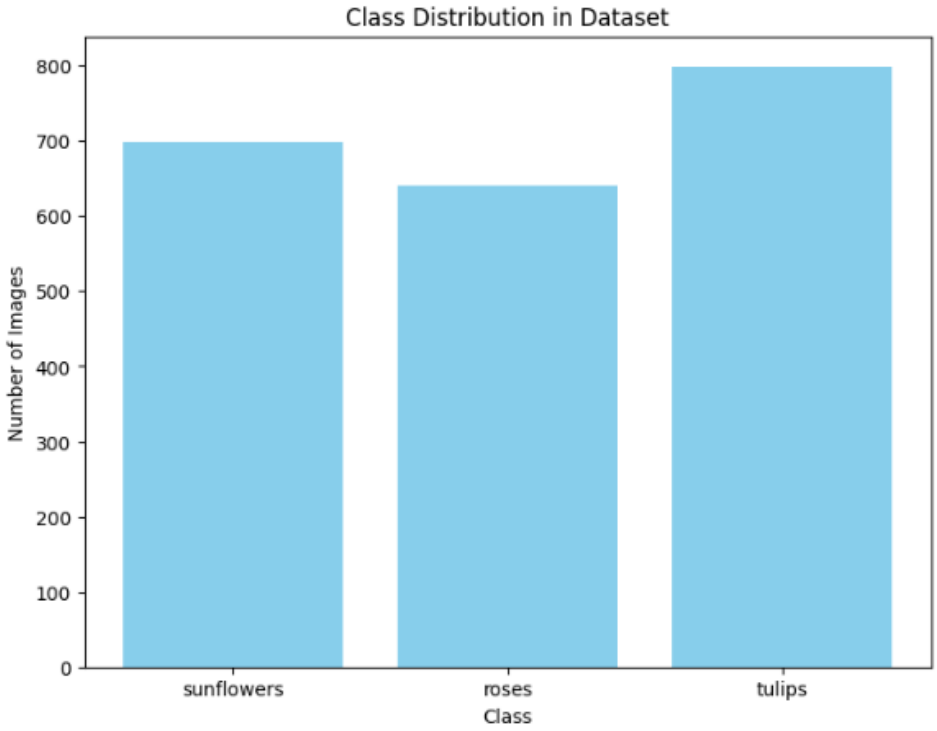
plt.title('Distribuição de Classes no Conjunto de Dados')

plt.show()
```

Co-funded by the European Union

Este projeto foi financiado com o apoio da Comissão Europeia. Esta comunicação reflete apenas as opiniões do autor, e a Comissão não pode ser responsabilizada por qualquer uso que possa ser feito da informação nela contida.

```
Class 'sunflowers': 699 images
Class 'roses': 641 images
Class 'tulips': 799 images
```




---

**Passo 4: AUMENTO DE DADOS PARA CLASSES DE EQUILÍBRIO**

O conjunto de dados que estamos a usar é algo desequilibrado, o que significa que certas classes de flores têm mais imagens do que outras, como mostrado no gráfico.

Um desequilíbrio pode afetar a aprendizagem do modelo, tornando mais provável que favoreça as classes com mais imagens, o que pode levar a previsões enviesadas.

Uma abordagem para lidar com este desequilíbrio seria eliminar imagens das classes sobrerrepresentadas para tornar os tamanhos dos conjuntos de dados iguais entre classes. No entanto, remover imagens reduziria a quantidade de informação de que o modelo pode aprender, o que seria uma perda em termos de qualidade e diversidade dos dados.

Em vez de apagar imagens, vamos usar aumento de dados para equilibrar o conjunto de dados. A ampliação de dados é uma técnica que gera novas imagens aplicando

Co-funded by the European Union

Este projeto foi financiado com o apoio da Comissão Europeia. Esta comunicação reflete apenas as opiniões do autor, e a Comissão não pode ser responsabilizada por qualquer uso que possa ser feito da informação nela contida.

transformações a imagens existentes. Isto permite-nos criar mais amostras das classes sub-representadas sem reunir novas imagens.

Transformações comuns usadas na ampliação de dados incluem:

Virar (horizontal ou vertical)

Rotação

Zoom

Ajustes de Cor

Para realizar aumento de dados no conjunto de dados, utilizaremos o ImageDataGenerator da TensorFlow para gerar e guardar imagens aumentadas para as classes sub-representadas. Este código aplicará transformações como rotações, flips e zooms para criar mais imagens, o que ajudará a equilibrar o conjunto de dados.

```
importar Tensorflow como TF

from tensorflow.keras.preprocessing.image import ImageDataGenerator

Importação de sistemas operativos



# Caminho para o diretório principal do conjunto de dados e número alvo de imagens por classe

data_dir = '/content/drive/My Drive/TrainingGreeceCV/datasets/flowers/'

target_num_images = 799  # Imagens desejadas por classe




# Definir transformações de aumento de dados
```

Co-funded by the European Union

Este projeto foi financiado com o apoio da Comissão Europeia. Esta comunicação reflete apenas as opiniões do autor, e a Comissão não pode ser responsabilizada por qualquer uso que possa ser feito da informação nela contida.

```
datagen = ImageDataGenerator(      # Inicializar uma instância do ImageDataGenerator para aumentar imagens

    rotation_range=40,             # Rodar aleatoriamente as imagens dentro de um intervalo de 40 graus para criar ângulos variados

    width_shift_range=0,2,         # Deslocar aleatoriamente as imagens horizontalmente até 20% da largura

    height_shift_range=0,2,        # Deslocar aleatoriamente as imagens verticalmente até 20% da altura

    shear_range=0,2,               # Aplicar transformações de cisalhamento aleatórias com um intervalo de 20% para variedade

    zoom_range=0,2,                # Amplie aleatoriamente as imagens até 20% para criar diferentes escalas

    horizontal_flip=Verdadeiro,          # Virar aleatoriamente as imagens horizontalmente para criar imagens espelhadas

    fill_mode='mais próximo'            # Preencha quaisquer píxeis recém-criados após transformações com os valores dos píxeis mais próximos

)



# Percorrer cada pasta de turma

Para class_name em os.listdir(data_dir):

    class_path = os.path.join(data_dir, class_name)



    # Verifica se isto é uma pasta e conta as imagens atuais

    if os.path.isdir(class_path):

        current_num_images = len(os.listdir(class_path))

        images_needed = target_num_images - current_num_images
```

Co-funded by the European Union

Este projeto foi financiado com o apoio da Comissão Europeia. Esta comunicação reflete apenas as opiniões do autor, e a Comissão não pode ser responsabilizada por qualquer uso que possa ser feito da informação nela contida.

```
        # Só aumentar se forem necessárias mais imagens
        se images_needed > 0:
            # Gerar e guardar imagens aumentadas na mesma pasta de
classe
            gerador = datagen.flow_from_directory(
                data_dir,
                classes=[class_name],
                target_size=(224, 224),
                batch_size=1,
                save_to_dir=class_path,  # Guardar imagens
aumentadas na mesma pasta das imagens originais
                save_prefix='aumentado',
                save_format='jpeg'
            )

            # Gerar o número exato de imagens necessárias
            para _ no intervalo(images_needed):
                próximo (gerador)
```

Agora, podemos executar o mesmo código de antes para aplicar aumento de dados e garantir que as classes estão equilibradas.

```
Importação de sistemas operativos
de coleções importação Contador
importar matplotlib.pyplot como plt
```

Co-funded by the European Union

Este projeto foi financiado com o apoio da Comissão Europeia. Esta comunicação reflete apenas as opiniões do autor, e a Comissão não pode ser responsabilizada por qualquer uso que possa ser feito da informação nela contida.

```
# Caminho para o diretório principal do conjunto de dados

data_dir = '/content/drive/My
Drive/TrainingGreeceCV/datasets/flowers/'



# Contar imagens em cada subdiretório (ou seja, cada classe)

class_counts = Contador()

Para class_name em os.listdir(data_dir):

    class_path = os.path.join(data_dir, class_name)

    if os.path.isdir(class_path):

 class_counts[class_name] = len(os.listdir(class_path))



# Traçar a distribuição de imagens por classe

plt.figure(figsize=(8, 6))

plt.bar(class_counts.keys(), class_counts.values(), color='azul-
celeste')

plt.xlabel('Classe')

plt.ylabel('Número de Imagens')

plt.title('Distribuição de Classes no Conjunto de Dados')

plt.show()
```

Co-funded by the European Union

Este projeto foi financiado com o apoio da Comissão Europeia. Esta comunicação reflete apenas as opiniões do autor, e a Comissão não pode ser responsabilizada por qualquer uso que possa ser feito da informação nela contida.




**Passo 5: ORGANIZAR O CONJUNTO DE DADOS PARA TREINO**

Organizar o Conjunto de Dados para Treino significa organizar os dados de forma estruturada para alinhar com os requisitos do modelo de aprendizagem automática. Diferentes modelos requerem diferentes estruturas de conjuntos de dados devido às suas diferentes arquiteturas e abordagens de treino.

Para este projeto, estamos a usar o MobileNetV3. A estrutura do MobileNet é relativamente simples comparada com outros modelos e requer uma estrutura de conjunto de dados direta.

O MobileNet espera que as imagens sejam organizadas por classe, onde cada classe tem a sua própria pasta contendo todas as suas imagens.

Exemplo da estrutura:

Co-funded by the European Union

Este projeto foi financiado com o apoio da Comissão Europeia. Esta comunicação reflete apenas as opiniões do autor, e a Comissão não pode ser responsabilizada por qualquer uso que possa ser feito da informação nela contida.

```
/dataset/
    ├── train/                  # 70% of data
    │   ├── class_1/            # Class 1 images for training
    │   │   ├── image1.jpg
    │   │   ├── image2.jpg
    │   │   └── ...
    │   ├── class_2/            # Class 2 images for training
    │   │   ├── image1.jpg
    │   │   ├── image2.jpg
    │   │   └── ...
    │   └── ...
    ├── validation/             # 15% of data
    │   ├── class_1/            # Class 1 images for validation
    │   │   ├── image1.jpg
    │   │   ├── image2.jpg
    │   │   └── ...
    │   ├── class_2/            # Class 2 images for validation
    │   │   ├── image1.jpg
    │   │   ├── image2.jpg
    │   │   └── ...
    │   └── ...
    ├── test/                   # 15% of data
    │   ├── class_1/            # Class 1 images for testing
    │   │   ├── image1.jpg
    │   │   ├── image2.jpg
    │   │   └── ...
    │   ├── class_2/            # Class 2 images for testing
    │   │   ├── image1.jpg
    │   │   ├── image2.jpg
    │   │   └── ...
    │   └── ...
```

No MobileNet, os nomes das pastas são usados para rotular automaticamente imagens, com cada pasta a representar uma classe específica. O modelo associa todas as imagens numa pasta com essa etiqueta de classe, eliminando a necessidade de ficheiros de etiqueta separados.

Dividir um conjunto de dados em conjuntos de treino, validação e teste ajuda o modelo a aprender padrões de uma parte dos dados, afinar os seus parâmetros e avaliar o seu desempenho em dados não vistos.

Co-funded by the European Union

Este projeto foi financiado com o apoio da Comissão Europeia. Esta comunicação reflete apenas as opiniões do autor, e a Comissão não pode ser responsabilizada por qualquer uso que possa ser feito da informação nela contida.

**Conjunto de Treino:** Esta parte, normalmente 70% dos dados, é usada pelo modelo para aprender padrões, características e relações em cada turma, fornecendo-lhe exemplos suficientes para construir uma compreensão robusta.

**Conjunto de Validação**: Cerca de 15% dos dados são usados para validação, permitindo ao modelo ajustar os seus parâmetros e evitar sobreajustes nos dados de treino. O conjunto de validação ajuda a ajustar o modelo sem influenciar a medição final de desempenho.

**Conjunto de Teste:** Os restantes 15% são guardados para teste. Estes dados não são observados pelo modelo durante o treino ou validação, permitindo uma avaliação imparcial do desempenho do modelo com novos dados. Um conjunto de teste de 15% fornece tipicamente uma estimativa fiável da precisão real do modelo.

Este processo é muito importante porque:

**Prevenção do Overfitting** Se o modelo treina e testa com os mesmos dados, pode "memorizar" o conjunto de dados em vez de aprender padrões que se aplicam a novos dados. Dividir os dados ajuda o modelo a aprender de forma a prepará-lo para novos exemplos.

**Avaliação Fiável** O conjunto de testes oferece uma forma de verificar a precisão do modelo e outras métricas em dados que ainda não viu. Isto ajuda a garantir que o desempenho do modelo é realista e não apenas bom com dados familiares.

**Melhor** Generalização O modelo treina com vários exemplos, mas é testado em dados separados. Esta configuração ajuda o modelo a aprender padrões que funcionam bem com novos dados, em vez de apenas nos exemplos de treino.

```
Importação de sistemas operativos

importar shutil

importar aleatório



# Defina o diretório de origem e os diretórios de destino para treinar
e testar

source_dir = '/conteúdo/drive/My Drive/Training/datasets/flowers/'

train_dir                    =                    '/content/drive/My
Drive/Training/datasets/flowersToTrain/train/'

test_dir                     =                    '/content/drive/My
Drive/Training/datasets/flowersToTrain/test/'
```

Co-funded by the European Union

Este projeto foi financiado com o apoio da Comissão Europeia. Esta comunicação reflete apenas as opiniões do autor, e a Comissão não pode ser responsabilizada por qualquer uso que possa ser feito da informação nela contida.

```
val_dir                    =                    '/content/drive/My
Drive/Training/datasets/flowersToTrain/val/'



# Definir razão de divisão

train_ratio = 0,7  # 70% para treino

test_ratio = 0,15   # 15% para testes

validation_ratio = 0,15   # 15% para validação




# Defina as proporções de divisão

train_ratio = 0,7

validation_ratio = 0,15

test_ratio = 0,15




# Criar diretórios-alvo se não existirem

os.makedirs(train_dir, exist_ok=True)

os.makedirs(val_dir, exist_ok=True)

os.makedirs(test_dir, exist_ok=True)



# Iterar por cada pasta de classes no diretório de origem

Para class_name em os.listdir(source_dir):

    class_path = os.path.join(source_dir, class_name)

    if os.path.isdir(class_path):
```

Co-funded by the European Union

Este projeto foi financiado com o apoio da Comissão Europeia. Esta comunicação reflete apenas as opiniões do autor, e a Comissão não pode ser responsabilizada por qualquer uso que possa ser feito da informação nela contida.

```
        # Criar subdiretórios em pastas de treino, validação e teste para cada classe
        os.makedirs(os.path.join(train_dir, class_name), exist_ok=True)
        os.makedirs(os.path.join(val_dir, class_name), exist_ok=True)
        os.makedirs(os.path.join(test_dir, class_name), exist_ok=True)

        # Obter uma lista de todas as imagens na pasta da turma e embaralhá-las
        imagens = os.list(class_path)
        random.shuffle(imagens)

        # Calcular contagens de divisões
 train_count = int(len(imagens) * train_ratio)
 validation_count = int(len(imagens) * validation_ratio)

        # Dividir imagens em conjuntos de comboio, validação e teste
        train_images = imagens[:train_count]
        validation_images = imagens[train_count:train_count + validation_count]
        test_images = imagens[train_count + validation_count:]

        # Mover imagens para a pasta do comboio
        Para o IMG em train_images:
            src_path = os.path.join(class_path, img)
```

Co-funded by the European Union

Este projeto foi financiado com o apoio da Comissão Europeia. Esta comunicação reflete apenas as opiniões do autor, e a Comissão não pode ser responsabilizada por qualquer uso que possa ser feito da informação nela contida.

```
            dst_path = os.path.join(train_dir, class_name, img)
            shutil.copy(src_path, dst_path)

        # Mover imagens para a pasta de validação
        Para o IMG em validation_images:
            src_path = os.path.join(class_path, img)
            dst_path = os.path.join(val_dir, class_name, img)
            shutil.copy(src_path, dst_path)

        # Mover imagens para a pasta de teste
        Para o IMG em test_images:
            src_path = os.path.join(class_path, img)
            dst_path = os.path.join(test_dir, class_name, img)
            shutil.copy(src_path, dst_path)

print("Conjunto de dados dividido com sucesso em conjuntos de treino, validação e teste.")
```

Verifique o número de ficheiros

```
def count_images(diretório):
    total_count = 0
    Para class_name no OS.listdir(diretório):
        class_folder = os.path.join(diretório, class_name)
        if os.path.isdir(class_folder):
            num_images = len(os.listdir(class_folder))
```

Co-funded by the European Union

Este projeto foi financiado com o apoio da Comissão Europeia. Esta comunicação reflete apenas as opiniões do autor, e a Comissão não pode ser responsabilizada por qualquer uso que possa ser feito da informação nela contida.

```
            total_count += num_images

            print(f"{class_name}    -    {directory.split('/')[-1]}:
{num_images} imagens")

    print(f"Total   imagens   em   {directory.split('/')[-1]}   set:
{total_count}\n")



print("\nContando imagens em cada conjunto:")

count_images(train_dir)

count_images(val_dir)

count_images(test_dir)
```

---

**Passo 6: Importar as bibliotecas**

Antes de construir e treinar o nosso modelo, precisamos de importar bibliotecas essenciais do TensorFlow e Keras. Estas bibliotecas fornecem ferramentas para pré-processamento de imagens, construção de modelos e otimização:

```
importar Tensorflow como TF

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.applications importar MobileNetV3Small

from tensorflow.keras.layers importar Dense, GlobalAveragePooling2D,
Dropout

from tensorflow.keras.models importar Modelo

from tensorflow.keras.optimizers importar Adam
```

**O que cada importação faz:**

- `tensorflow` – Framework central de deep learning utilizado ao longo deste projeto.

Co-funded by the European Union

Este projeto foi financiado com o apoio da Comissão Europeia. Esta comunicação reflete apenas as opiniões do autor, e a Comissão não pode ser responsabilizada por qualquer uso que possa ser feito da informação nela contida.

- `ImageDataGenerator` – Ajuda a carregar e aumentar imagens em lotes durante o treino.
- `MobileNetV3Small` – Um modelo leve, pré-treinado, que vamos adaptar para classificação de flores.
- `Dense`, `GlobalAveragePooling2D`, `Dropout` – Camadas usadas para construir o classificador personalizado.
- `Modelo` – Permite-nos definir um modelo que combina a base MobileNet com as nossas novas camadas.
- `Adam` – Um otimizador adaptativo usado para treinar a rede de forma eficiente.

---

**Passo 7: Configurar Geradores de Dados**

Configurar geradores de dados é essencial para treinar modelos de aprendizagem automática por várias razões.

Em primeiro lugar, os geradores de dados gerem o carregamento e a memória eficientes ao tratar imagens em lotes, o que previne o excesso de memória. Em vez de carregar todas as imagens de uma só vez, os geradores de dados carregam apenas algumas imagens de cada vez, permitindo treinar em grandes conjuntos de dados sem usar demasiada memória.

Em segundo lugar, os geradores de dados permitem a ampliação de dados em tempo real, o que ajuda a melhorar a robustez dos modelos, expandindo o conjunto de dados através de transformações como rotações, deslocamentos, zooms e flips. Isto significa que o modelo vê versões ligeiramente diferentes de cada imagem durante cada época de treino, o que ajuda a reduzir o sobreajuste e melhora a capacidade do modelo de lidar com variações em novos dados.

Em terceiro lugar, os geradores de dados atribuem automaticamente etiquetas e realizam pré-processamento básico, como redimensionar e normalizar imagens, com base na estrutura da pasta.

```
# Definir diretórios para conjuntos de dados de treino e validação
```

Co-funded by the European Union

Este projeto foi financiado com o apoio da Comissão Europeia. Esta comunicação reflete apenas as opiniões do autor, e a Comissão não pode ser responsabilizada por qualquer uso que possa ser feito da informação nela contida.

```
train_dir                    =                    '/content/drive/My
Drive/Training/datasets/flowersToTrain/train/'

val_dir                      =                    '/content/drive/My
Drive/Training/datasets/flowersToTrain/val/'




# Configurar o ImageDataGenerators para treino e teste com aumento
de dados

train_datagen = ImageDataGenerator(rotation_range=15)

train_generator    =    train_datagen.flow_from_directory(train_dir,
target_size=(224, 224), batch_size=32)




val_datagen = ImageDataGenerator(rotation_range=15)

val_generator       =       val_datagen.flow_from_directory(val_dir,
target_size=(224, 224), batch_size=32)
```

O número de imagens criadas nesta fase através de aumento de dados depende do número de épocas e lotes usados durante o treino, em vez de uma contagem fixa.

A ampliação de dados gera novas variações de imagens em cada lote, produzindo um fornecimento "infinito" de dados de treino sem realmente aumentar o tamanho físico do conjunto de dados. Ao usar um ImageDataGenerator com aumentação, cada lote fornece uma "vista" modificada das imagens originais, com transformações como rotação, zoom e deslocamentos. Isto significa que o número total de variações únicas de imagem que o modelo encontra é determinado pelo número de épocas e lotes.

Estas aumentações devem ser usadas com cuidado, pois transformações excessivas ou inadequadas podem distorcer as imagens, levando a uma má aprendizagem dos modelos. Por exemplo, rotações extremas, zooms ou flips podem criar versões

Co-funded by the European Union

Este projeto foi financiado com o apoio da Comissão Europeia. Esta comunicação reflete apenas as opiniões do autor, e a Comissão não pode ser responsabilizada por qualquer uso que possa ser feito da informação nela contida.

irrealistas de imagens que não representam as classes reais, confundindo o modelo em vez de o melhorar. Além disso, se certas melhorias (como flips) forem aplicadas a classes onde a orientação importa, o modelo pode ter dificuldades em distinguir entre orientações corretas e incorretas, reduzindo a precisão. Aumentos devidamente equilibrados ajudam o modelo a generalizar ao introduzir variações úteis, mas o uso excessivo ou a má aplicação podem afetar negativamente o seu desempenho em dados do mundo real.

---

**Passo 8: Carregar o MobileNetV3 pré-treinado e modificar para aprendizagem por transferência**

Vamos carregar o modelo MobileNetV3, remover a sua camada final de classificação e adicionar novas camadas para as nossas classes personalizadas. Congele as camadas pré-treinadas para manter as características aprendidas e adicione uma camada de saída personalizada para as classes específicas do nosso conjunto de dados.

Este código seguinte cria um modelo de classificação personalizado baseado no MobileNetV3Small, um modelo pré-treinado, que é usado como extrator de características ao definir include_top=False para remover a sua camada original de classificação.

O modelo base produz mapas de características, que são depois passados por uma série de camadas personalizadas: uma camada global de pooling médio para reduzir a dimensionalidade e três camadas totalmente ligadas com ativação ReLU para ajudar o modelo a aprender padrões complexos.

A camada final de saída tem 3 unidades com ativação softmax, permitindo a classificação multi-classe para três categorias. O modelo é configurado ligando a entrada do modelo base a esta nova camada de saída.

Todas as camadas, exceto as últimas cinco, estão congeladas, por isso apenas as novas camadas são atualizadas durante o treino, o que ajuda a manter as funcionalidades aprendidas do MobileNet e reduz a probabilidade de overfitting.

```
baseModel = MobileNetV3Small(weights="imagenet", include_top=False,
input_shape=(224, 224, 3))



# Adicionar camadas personalizadas por cima do modelo base
```

Co-funded by the European Union

Este projeto foi financiado com o apoio da Comissão Europeia. Esta comunicação reflete apenas as opiniões do autor, e a Comissão não pode ser responsabilizada por qualquer uso que possa ser feito da informação nela contida.

```
x = baseModel.output

x = GlobalAveragePooling2D()(x)                 # Adicionar a camada
global de média de pooling

x = Denso(512, ativação='relu')(x)               # Adicionar camada
totalmente ligada com 512 unidades

x = Denso(256, ativação='relu')(x)               # Adicionar camada
totalmente ligada com 256 unidades

x = Denso(128, ativação='relu')(x)               # Adicionar camada
totalmente ligada com 128 unidades



# Camada de saída com 3 classes e ativação softmax para classificação
multi-classe

Camada de previsão = Denso(3, ativação='softmax')(x)



# Defina o modelo completo ligando a entrada do modelo base à camada
de previsão personalizada

model = Model(inputs=baseModel.input, outputs=Camada de previsão)



# Congelar as camadas do MobileNetV3 (camadas já treinadas)

para a camada em model.layers[:-5]:   # Congelar todas as camadas
exceto as últimas 5

    layer.trainable = Falso
```

---

**Passo 9: Compilar o Modelo**

Co-funded by the European Union

Este projeto foi financiado com o apoio da Comissão Europeia. Esta comunicação reflete apenas as opiniões do autor, e a Comissão não pode ser responsabilizada por qualquer uso que possa ser feito da informação nela contida.

Para preparar o modelo para treino, precisamos de o compilar especificando um otimizador, uma função de perda e métricas de avaliação.

**Otimizador**: O otimizador ajusta os pesos do modelo durante o treino para minimizar a função de perda. Aqui, usamos o otimizador Adam, uma escolha popular para modelos de deep learning, pois adapta a taxa de aprendizagem de cada parâmetro individualmente, tornando o treino eficiente. Para a aprendizagem por transferência, definimos uma taxa de aprendizagem mais pequena que permite o ajuste fino das novas camadas adicionadas sobre o modelo pré-treinado sem fazer alterações drásticas nos pesos pré-treinados. Esta abordagem ajuda a manter as funcionalidades benéficas aprendidas com o conjunto de dados original do ImageNet, adaptando o modelo a novos dados.

**Função de Perda:** A função de perda mede quão bem as previsões do modelo correspondem aos rótulos verdadeiros. Usamos a crossentropia categórica, que compara a distribuição de probabilidade prevista entre classes com a distribuição verdadeira.

**Métricas:** As métricas são usadas para avaliar o desempenho do modelo durante o treino e os testes. Usamos a precisão como métrica principal para tarefas de classificação, pois indica a percentagem de imagens corretamente classificadas.

```
# Compilar o modelo com otimizador, função de perda e métricas

otimizador = Adam(learning_rate=0,0001)

modelo.compilar                    (loss="categorical_crossentropy",
otimizador=otimizador, métricas=['precisão'])
```

---

**Passo 10: Treinar e guardar o modelo**

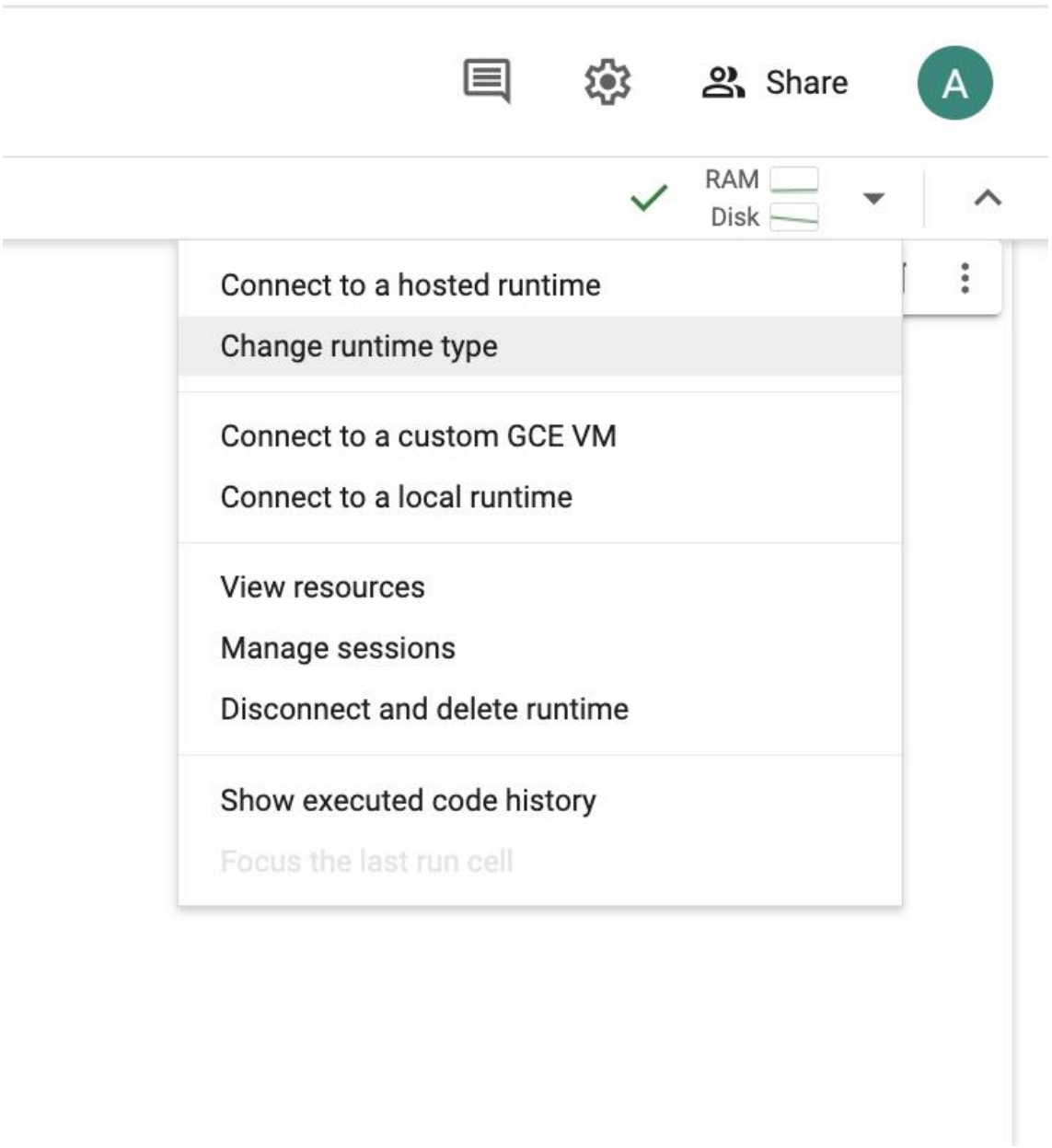
No ambiente Colab, certifica-te de que tens acesso a uma GPU T4 (Unidade de Processamento Gráfico). A GPU T4 é usada para aprendizagem profunda e treino de modelos porque consegue lidar com os cálculos paralelos necessários para tarefas como operações matriciais e manipulações tensoriais, que são comuns em redes neuronais.

Para mudar de CPU para GPU no Google Colab, siga estes passos:

1- No seu caderno Colab, vá ao menu superior e selecione Runtime e selecione Alterar tipo de runtime.

Co-funded by the European Union

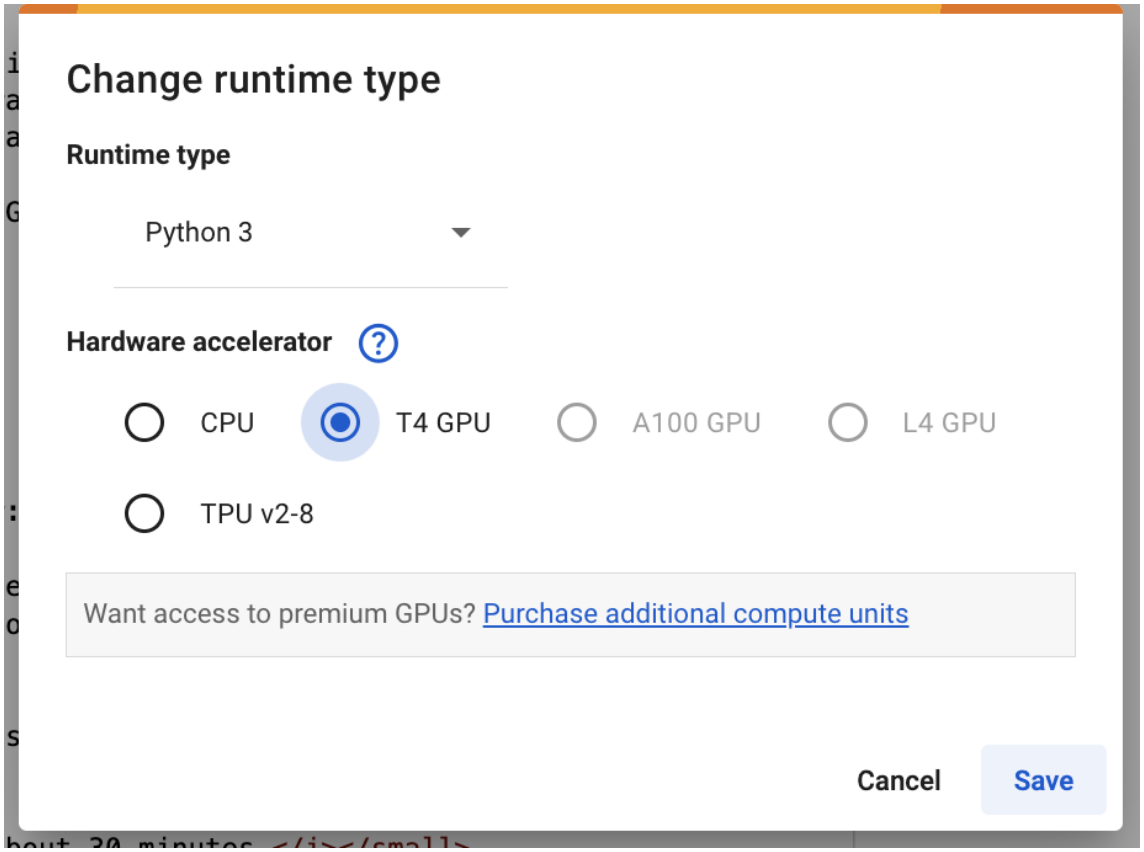
Este projeto foi financiado com o apoio da Comissão Europeia. Esta comunicação reflete apenas as opiniões do autor, e a Comissão não pode ser responsabilizada por qualquer uso que possa ser feito da informação nela contida.




2- Na janela pop-up, encontre o menu suspenso do acelerador de hardware. Selecione GPU nas opções suspensas (pode também ver Nenhum e TPU como outras opções). Clique em Guardar para confirmar a sua escolha.

Co-funded by the European Union

Este projeto foi financiado com o apoio da Comissão Europeia. Esta comunicação reflete apenas as opiniões do autor, e a Comissão não pode ser responsabilizada por qualquer uso que possa ser feito da informação nela contida.




Esta tarefa demora cerca de 30 minutos.

```
model.fit(train_generator,            validation_data=val_generator,
epochs=10)


model.save('/content/drive/My
Drive/TrainingGreeceCV/models/flower_classifier_model_AUG_FV2.keras
')
```

---

**Passo 11: Avaliar o modelo**

O código seguinte avalia o nosso modelo no conjunto de testes, carregando os dados de teste, prevendo as classes e calculando várias métricas de desempenho.

Os dados de teste são carregados com o ImageDataGenerator num formato não aumentado, que fornece imagens ao modelo sem embaralhar para garantir resultados consistentes.

Co-funded by the European Union

Este projeto foi financiado com o apoio da Comissão Europeia. Esta comunicação reflete apenas as opiniões do autor, e a Comissão não pode ser responsabilizada por qualquer uso que possa ser feito da informação nela contida.

A etapa de avaliação do modelo fornece métricas de perda e precisão, onde a perda reflete o erro de previsão, e a precisão mostra a proporção de previsões corretas em relação ao total das amostras.

- **A matriz de confusão** mostra visualmente o desempenho do modelo em cada classe, mostrando com que frequência o modelo previu corretamente ou incorretamente cada classe, o que ajuda a identificar padrões de má classificação.
- **A precisão** mede quantos dos resultados positivos previstos são realmente positivos (útil para reduzir falsos positivos)
- **A** recordação mede quantos positivos reais foram corretamente identificados (útil para reduzir falsos negativos)
- **A pontuação F1** combina precisão e recordação para fornecer uma medida equilibrada, especialmente útil para conjuntos de dados desequilibrados.

O relatório de classificação apresenta um resumo por classe da precisão, recordação e pontuação F1, fornecendo uma visão mais profunda do desempenho dos modelos para cada classe.

- **A curva ROC** (Receiver Operating Characteristic curve) representa a taxa de verdadeiros positivos contra a taxa de falsos positivos, ajudando a avaliar a capacidade do modelo para distinguir entre classes, com a pontuação AUC (Área Sob a Curva) a indicar o desempenho global da classificação — valores mais próximos de 1,0 sugerem melhor distinção entre classes.

```
importar numpy como np

importar matplotlib.pyplot como plt

Importação Seaborn como SNS

Do sklearn.metrics importa confusion_matrix

de tensorflow.keras.models importar load_model

from tensorflow.keras.preprocessing.image import ImageDataGenerator


Do sklearn.metrics importa precision_score, recall_score, f1_score,
roc_curve, AUC, classification_report

importar matplotlib.pyplot como plt

importar numpy como np



model = load_model('/content/drive/My
Drive/Training/models/flower_classifier_model_AUG_FV2.keras')
```

Co-funded by the European Union

Este projeto foi financiado com o apoio da Comissão Europeia. Esta comunicação reflete apenas as opiniões do autor, e a Comissão não pode ser responsabilizada por qualquer uso que possa ser feito da informação nela contida.

```
# Configurar o ImageDataGenerator para o conjunto de testes
# Definir o diretório de teste
test_dir                    =                    '/content/drive/My
Drive/Training/datasets/flowersToTrain/test/'

# Configurar o ImageDataGenerator para os dados de teste sem aumento
test_datagen = ImageDataGenerator()  # Sem aumento de dados

# Criar o gerador de testes
test_generator = test_datagen.flow_de_diretório(
    test_dir,
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical',   # Muda para 'binário' se for uma
tarefa de classificação binária
    shuffle=False  # Não embaralhes dados do teste
)

# Calcular o número de passos para avaliação com base no gerador de
testes
evaluation_steps = test_generator.samples // test_generator.tamanho
do lote

# Avaliar o modelo para confirmar perda e precisão
perda, precisão = model.avaliar(test_generator)
print(f"Test Loss: {loss:.4f}")
print(f"Teste de Precisão: {Precisão:.4f}")

# Gerar previsões
```

Co-funded by the European Union

Este projeto foi financiado com o apoio da Comissão Europeia. Esta comunicação reflete apenas as opiniões do autor, e a Comissão não pode ser responsabilizada por qualquer uso que possa ser feito da informação nela contida.

```
previsões = model.predict(test_generator)

y_pred = np.argmax(previsões, eixo=1)

y_true = test_generator.classes

y_pred_prob = previsões se previsões.shape[1] == 2 outras previsões[:, 1]  # Para ROC


# Matriz de confusão

cm = confusion_matrix(y_true, y_pred)

class_names = list(test_generator.class_indices.keys())


# Traçar a matriz de confusão

plt.figure(figsize=(10, 8))

sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=class_names, yticklabels=class_names)

plt.xlabel('Etiqueta Prevista')

plt.ylabel ('Etiqueta Verdadeira')

plt.title('Matriz de Confusão')

plt.show()


# Calcular precisão, recordação e pontuações F1

Precisão = precision_score(y_true, y_pred, média='ponderado')

recordação = recall_score(y_true, y_pred, média = 'ponderado')

f1 = f1_score(y_true, y_pred, média='ponderado')


# Relatório de classificação impressa

class_names = list(test_generator.class_indices.keys())

relatório = classification_report(y_true, y_pred, target_names=class_names)

print ("Relatório de Classificação:\n", relatório)

print(f"Precisão: {precisão:.4f}, Recall: {recall:.4f}, F1 Pontuação: {f1:.4f}")
```

Co-funded by the European Union

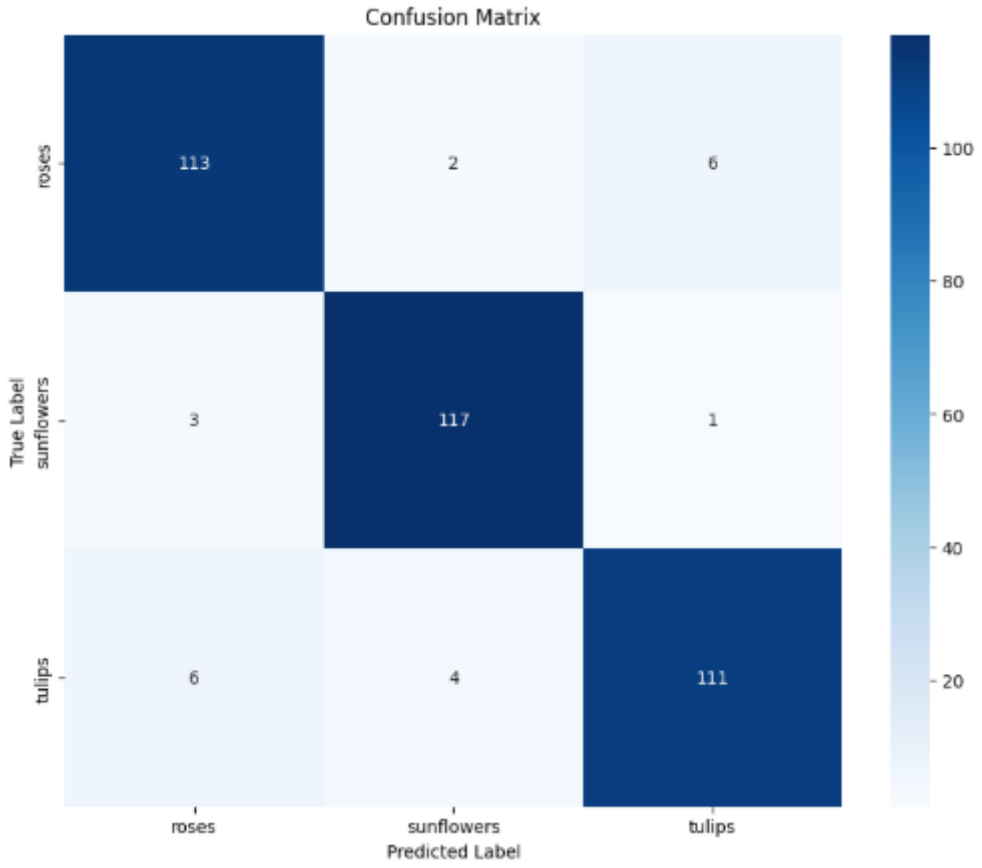
Este projeto foi financiado com o apoio da Comissão Europeia. Esta comunicação reflete apenas as opiniões do autor, e a Comissão não pode ser responsabilizada por qualquer uso que possa ser feito da informação nela contida.

```
# Traçado da Curva ROC
plt.figure(figsize=(8, 6))
if len(class_names) == 2:  # Classificação binária
    fpr, tpr, _ = roc_curve(y_true, y_pred_prob)
    roc_auc = auc(fpr, tpr)
    plt.plot(fpr, tpr, color='blue', label=curva f'ROC (área = {roc_auc:.2f})')
else else: # Classificação multiclasse
    Para I no alcance (len(class_names)):
        FPR, TPR, _ = roc_curve(y_true == i, previsões[:, i])
        roc_auc = auc(fpr, tpr)
        plt.plot(fpr, tpr, label=curva f'ROC da classe {class_names[i]} (área = {roc_auc:.2f})')

plt.plot([0, 1], [0, 1], color='gray', linestyle='--')
plt.xlabel ('Taxa de Falsos Positivos')
plt.ylabel ('Taxa de Verdadeiros Positivos')
plt.title('Curva da Característica de Funcionamento do Recetor (ROC)')
plt.legend(loc='canto inferior direito')
plt.show()
```

Co-funded by the European Union

Este projeto foi financiado com o apoio da Comissão Europeia. Esta comunicação reflete apenas as opiniões do autor, e a Comissão não pode ser responsabilizada por qualquer uso que possa ser feito da informação nela contida.




Co-funded by the European Union

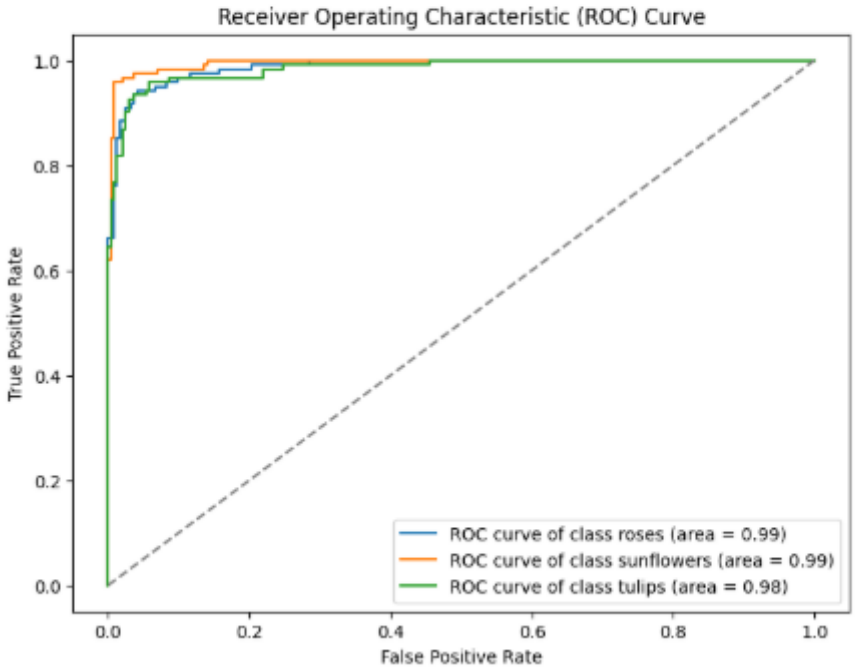
Este projeto foi financiado com o apoio da Comissão Europeia. Esta comunicação reflete apenas as opiniões do autor, e a Comissão não pode ser responsabilizada por qualquer uso que possa ser feito da informação nela contida.

```
Classification Report:
              precision    recall  f1-score   support

       roses       0.93      0.93      0.93       121
  sunflowers       0.95      0.97      0.96       121
      tulips       0.94      0.92      0.93       121

    accuracy                           0.94       363
   macro avg       0.94      0.94      0.94       363
weighted avg       0.94      0.94      0.94       363

Precision: 0.9394, Recall: 0.9394, F1 Score: 0.9393
```




**Passo 12 :P redito sobre nova imagem**

Depois de treinar e guardar o nosso modelo de classificação de flores, podemos usá-lo para prever a classe de uma nova imagem não vista. Este é um passo importante para verificar se o modelo funciona corretamente fora dos conjuntos de treino e validação.

Como Obter a Imagem

Escolha qualquer imagem de uma flor (rosa, tulipa ou girassol) do seu computador ou de uma fonte online como o Google Imagens.

Renomeie a imagem (por exemplo, t.jpeg) e faça o upload manualmente para o Google Drive em: /Training/

Co-funded by the European Union

Este projeto foi financiado com o apoio da Comissão Europeia. Esta comunicação reflete apenas as opiniões do autor, e a Comissão não pode ser responsabilizada por qualquer uso que possa ser feito da informação nela contida.

Certifique-se de que é uma fotografia do mundo real — isto ajuda a avaliar quão bem o modelo generaliza.

```
do tensorflow.keras.preprocessing.image import load_img, img_to_array

importar numpy como np



model = load_model('/content/drive/My Drive/Training/models/flower_classifier_model_AUG_FV2.keras')

# Configurar o ImageDataGenerators para treino e teste com aumento de dados



train_dir = '/content/drive/My Drive/Training/datasets/flowersToTrain/train/'

train_datagen = ImageDataGenerator()

train_generator = train_datagen.flow_from_directory(train_dir, target_size=(224, 224), batch_size=32)

train_dir = '/content/drive/My Drive/Training/datasets/flowersToTrain/train/'









# Caminho para a imagem única

img_path = '/content/drive/My Drive/Training/t.jpeg'



# Carregar e pré-processar a imagem

img = load_img(img_path, target_size=(224, 224))

img_array = img_to_array(img)
```

Co-funded by the European Union

Este projeto foi financiado com o apoio da Comissão Europeia. Esta comunicação reflete apenas as opiniões do autor, e a Comissão não pode ser responsabilizada por qualquer uso que possa ser feito da informação nela contida.

```
img_array = np.expand_dims(img_array, eixo=0)  # Adicionar dimensão em lote


# Prever a turma

previsão = model.predict(img_array)

predicted_class = np.argmax(previsão, eixo=1)[0]

Imprimir (previsão)


# Índice do mapa para o rótulo da classe

class_labels = {v: k para k, v em train_generator.class_indices.items()}


# Imprimir previsões para todas as turmas com percentagens

print("Previsões para todas as classes:")

para o índice, probabilidade em enumerar(previsão[0]):

    Etiqueta = class_labels[Índice]

    percentagem = probabilidade * 100  # Converter para percentagem

    print(f"Class: {label}, Probabilidade: {percentage:.2f}%")


# Imprimir a melhor classe prevista

predicted_class = np.argmax(previsão, eixo=1)[0]

predicted_label = class_labels[predicted_class]

predicted_probability = previsão[0][predicted_class] * 100

print(f"\nClasse prevista: {predicted_label} com confiança: {predicted_probability:.2f}%")
```

Co-funded by the European Union

Este projeto foi financiado com o apoio da Comissão Europeia. Esta comunicação reflete apenas as opiniões do autor, e a Comissão não pode ser responsabilizada por qualquer uso que possa ser feito da informação nela contida.

```
Found 1677 images belonging to 3 classes.
1/1 ━━━━━━━━━━━━━━━━━━━━ 3s 3s/step
[[7.7244986e-05 1.1240940e-05 9.9991155e-01]]
Predictions for all classes:
Class: roses, Probability: 0.01%
Class: sunflowers, Probability: 0.00%
Class: tulips, Probability: 99.99%

Predicted class: tulips with confidence: 99.99%
```

---

**11. Critérios de Avaliação:**

**Configuração do Conjunto** de Dados: Imagens recolhidas, rotuladas e organizadas corretamente

**Aumento de Dados**: Classes sub-representadas aumentadas usando o ImageDataGenerator

**Integração com MobileNet**: O modelo utiliza MobileNet com aprendizagem por transferência

**Precisão e Avaliação**: Matriz de confusão, precisão, recordação, pontuação F1 reportada

**Previsão final**: O modelo faz previsões sobre novas imagens |

---

**12. Línguas Disponíveis:**

**Inglês**, **Português** (traduções em progresso)

---

**13. Colaboradores:**

**Fátima Pais, Luís Soares** – Designers e Facilitadores de Projetos

Co-funded by the European Union

Este projeto foi financiado com o apoio da Comissão Europeia. Esta comunicação reflete apenas as opiniões do autor, e a Comissão não pode ser responsabilizada por qualquer uso que possa ser feito da informação nela contida.

**14. Feedback e Melhoria Contínua:**

- **Inquéritos**: No final deste marco, alunos e educadores irão preencher inquéritos de feedback para avaliar a eficácia do projeto.
- **Revisões de Projetos**: Após a conclusão do projeto, será realizada uma reunião final de revisão para discutir potenciais melhorias para futuros projetos baseados em IA.
- **Análise de Dados**: Os dados de feedback serão analisados para identificar pontos fortes e áreas a melhorar, informando projetos futuros.

Co-funded by the European Union

Este projeto foi financiado com o apoio da Comissão Europeia. Esta comunicação reflete apenas as opiniões do autor, e a Comissão não pode ser responsabilizada por qualquer uso que possa ser feito da informação nela contida.