

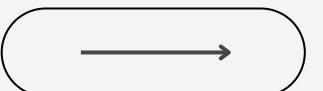
JORGE RICARDO MARQUES DUARTE  
a202110042@isec.pt

DATE  
18/12/2023



# **IDENTIFICAÇÃO DA SAÚDE DE PRODUTOS VEGETAIS**

Fase 3 - Inteligência computacional



# Introdução



A identificação precoce e eficaz de doenças em plantas emerge como uma solução vital no âmbito do desenvolvimento sustentável.

# TÓPICOS

01

Ajustes no dataset (%)

02

Seleção de uma arquitetura pré-treinada

03

Definição de uma rede densa para as últimas camadas, e otimização dos hiper-parâmetros

04

Análise dos resultados

05

Criação do melhor modelo e treino fazendo uso dos melhores hiperparâmetros

06

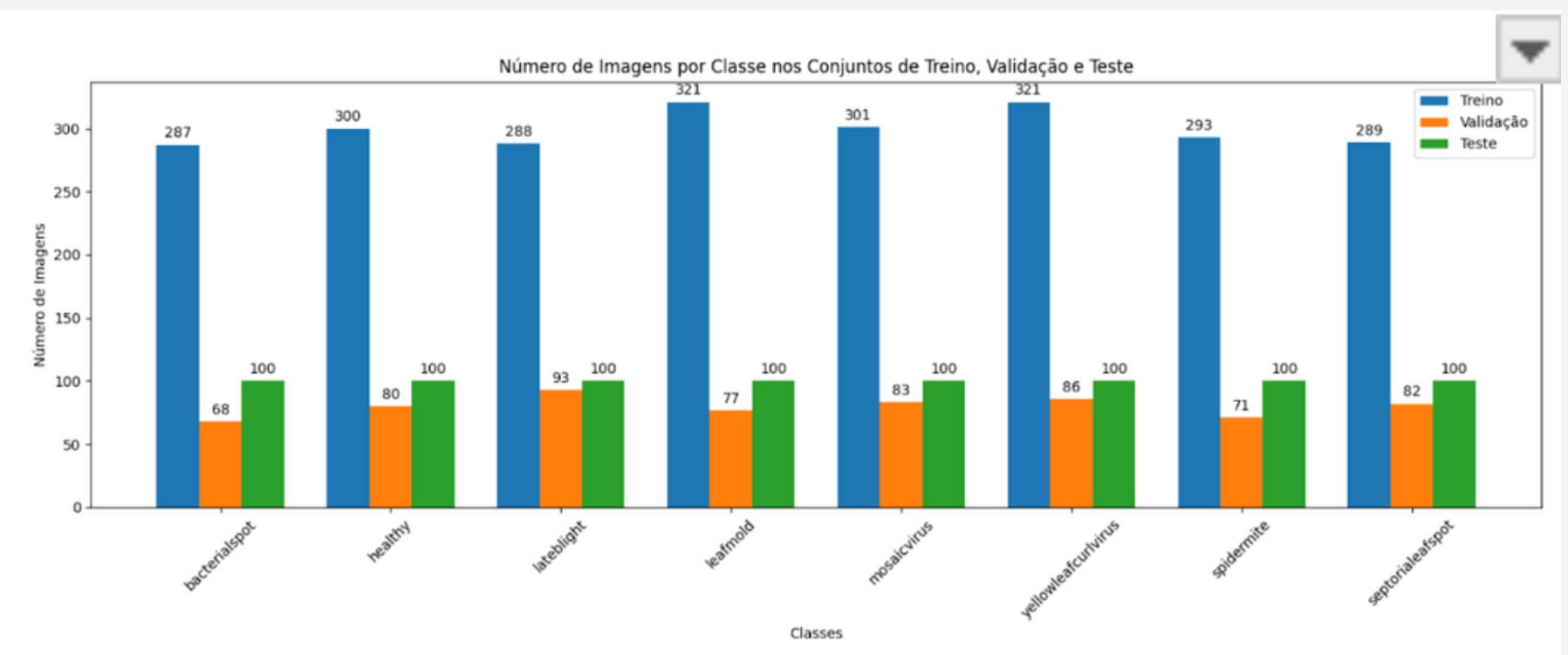
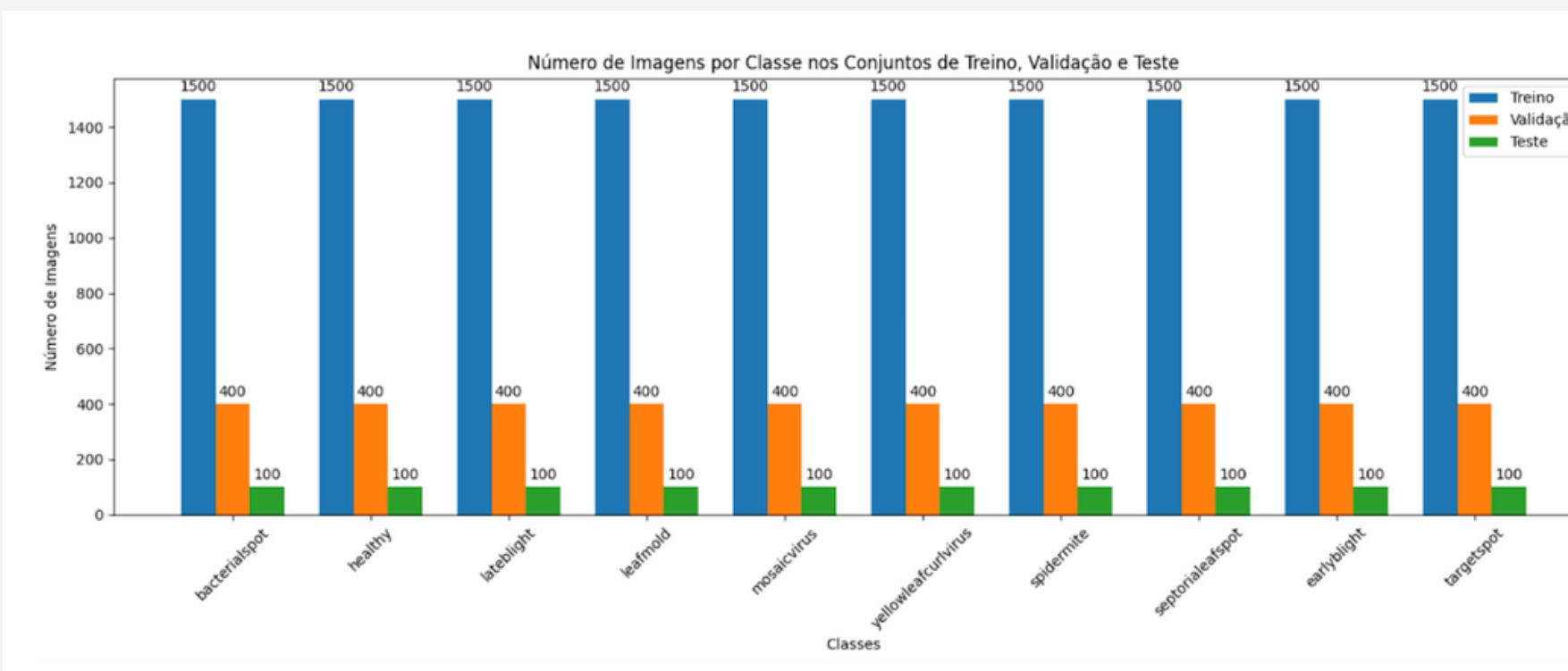
Aplicação real do modelo

07

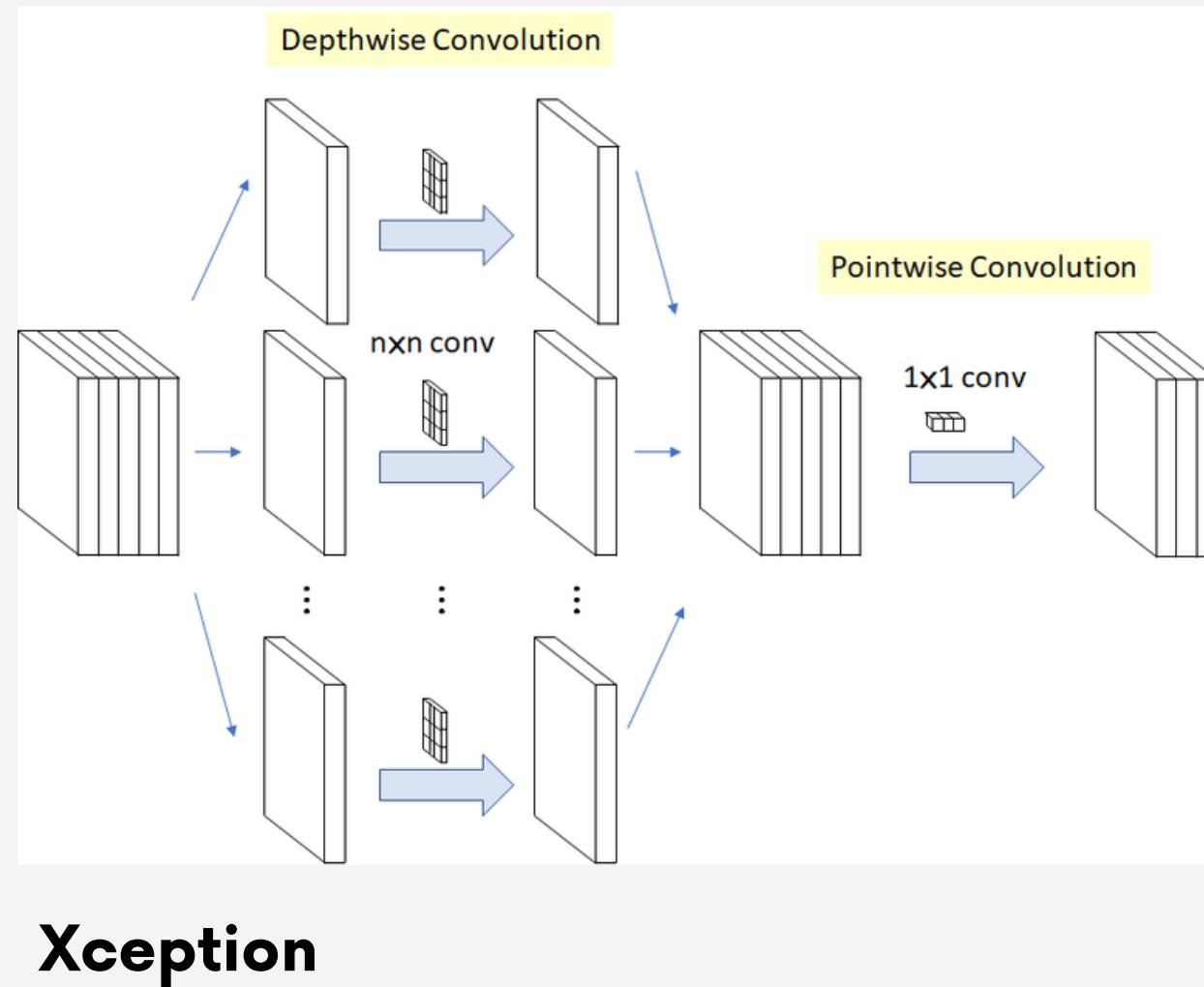
Q/A

Redução de 80% para o conjunto de teste e validação para identificar os melhores hiperparametros.

# DATASET

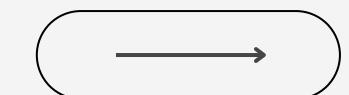
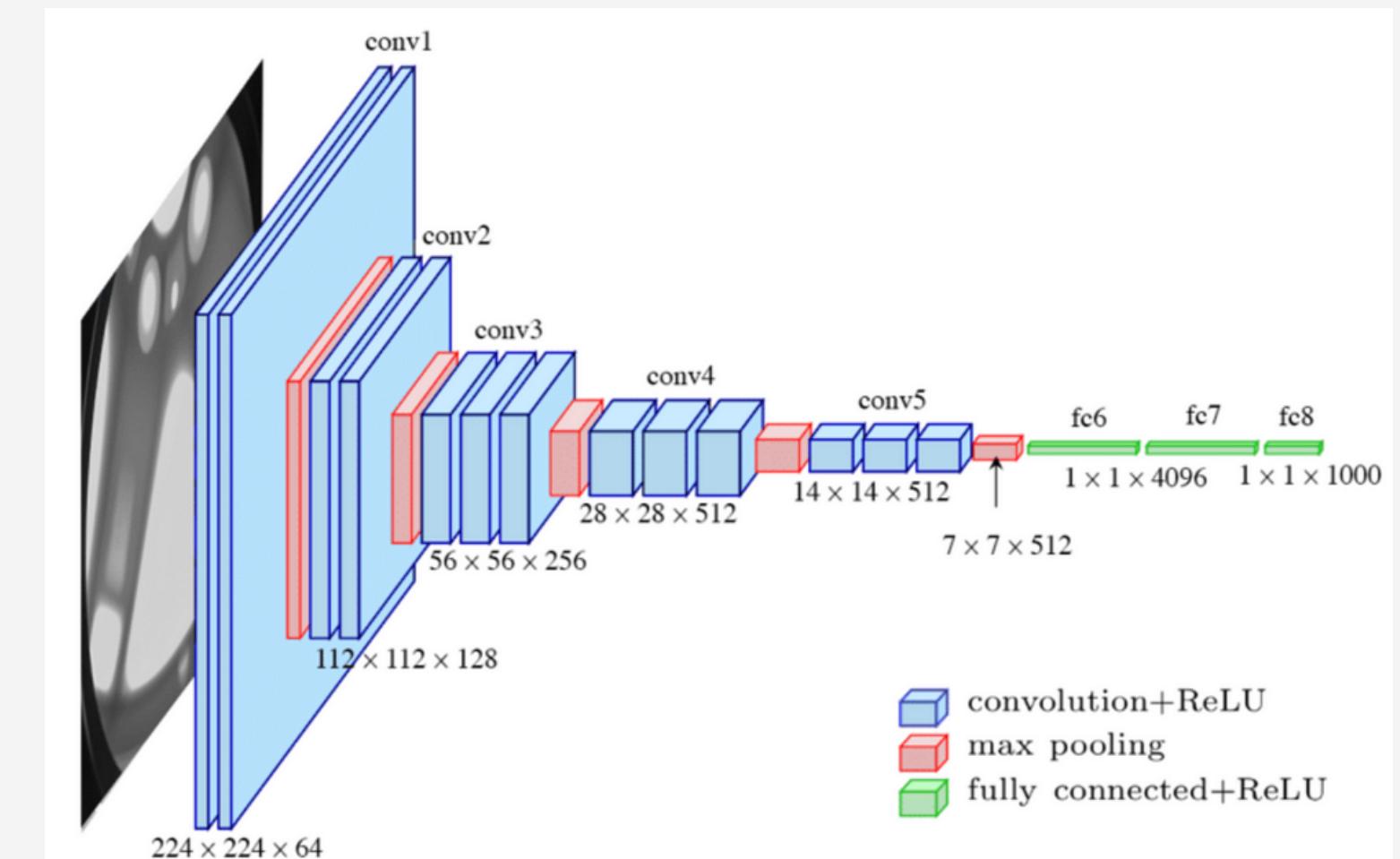


02



# ARQUITETURAS

Para a realização do trabalho, foram feito testes com a arquitetura VGG-16 e a Xception.



# DEFINIÇÃO DA REDE DENSA

## TOP LAYERS

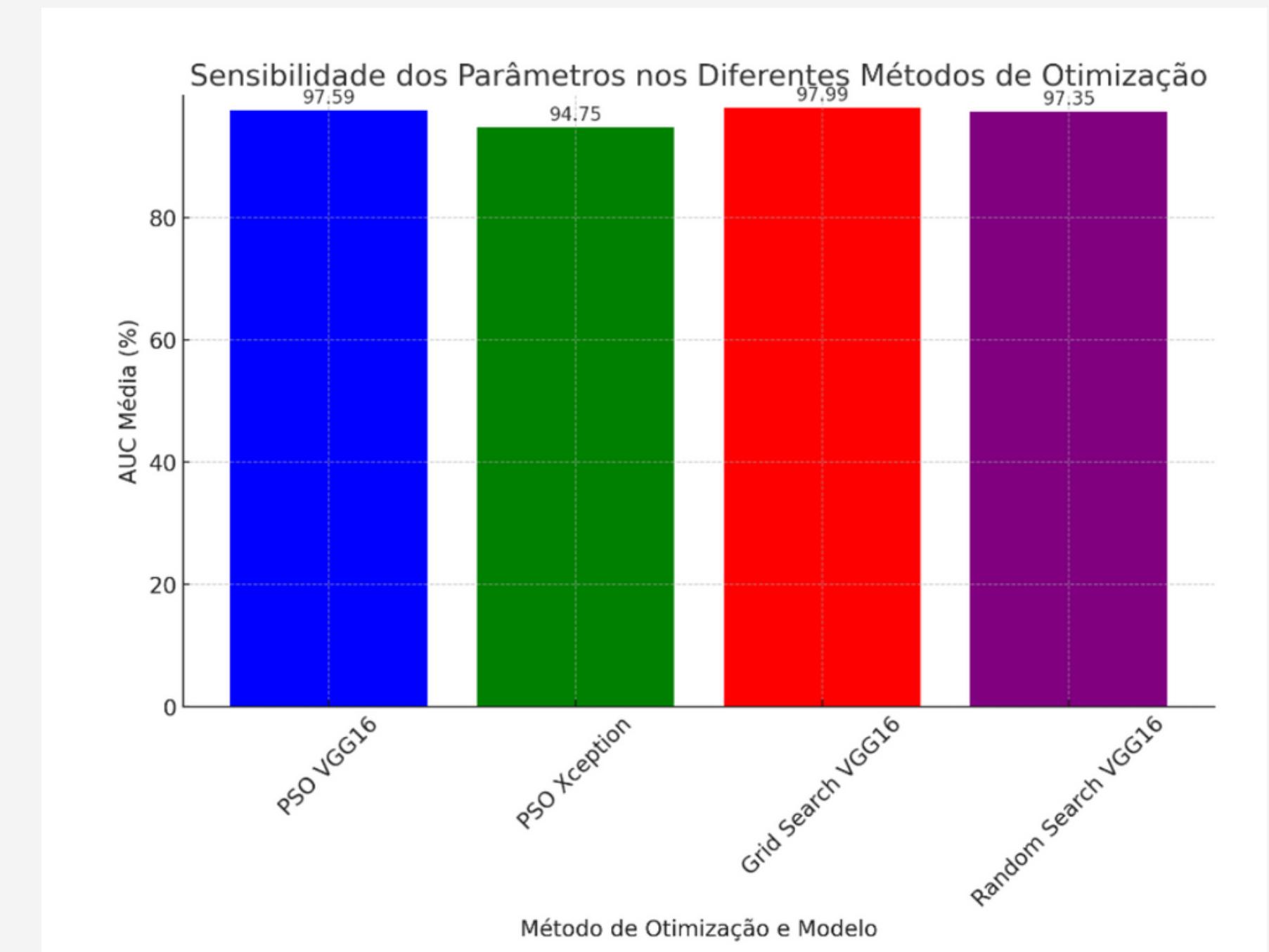
Flatten	Achatamento da saída do modelo base
Dense	128 neurônios, ativação ReLU.
Dropout	Taxa de dropout definida pelo hiperparâmetro
Dense	Número de neurônios igual ao número de classes, ativação softmax.

03

# OTIMIZAÇÃO DOS HIPER- PARÂMETROS

Função utilizada para a criação dos modelos baseado nos hiper-parâmetros dropout e learning rate.

# ANÁLISE DE RESULTADOS



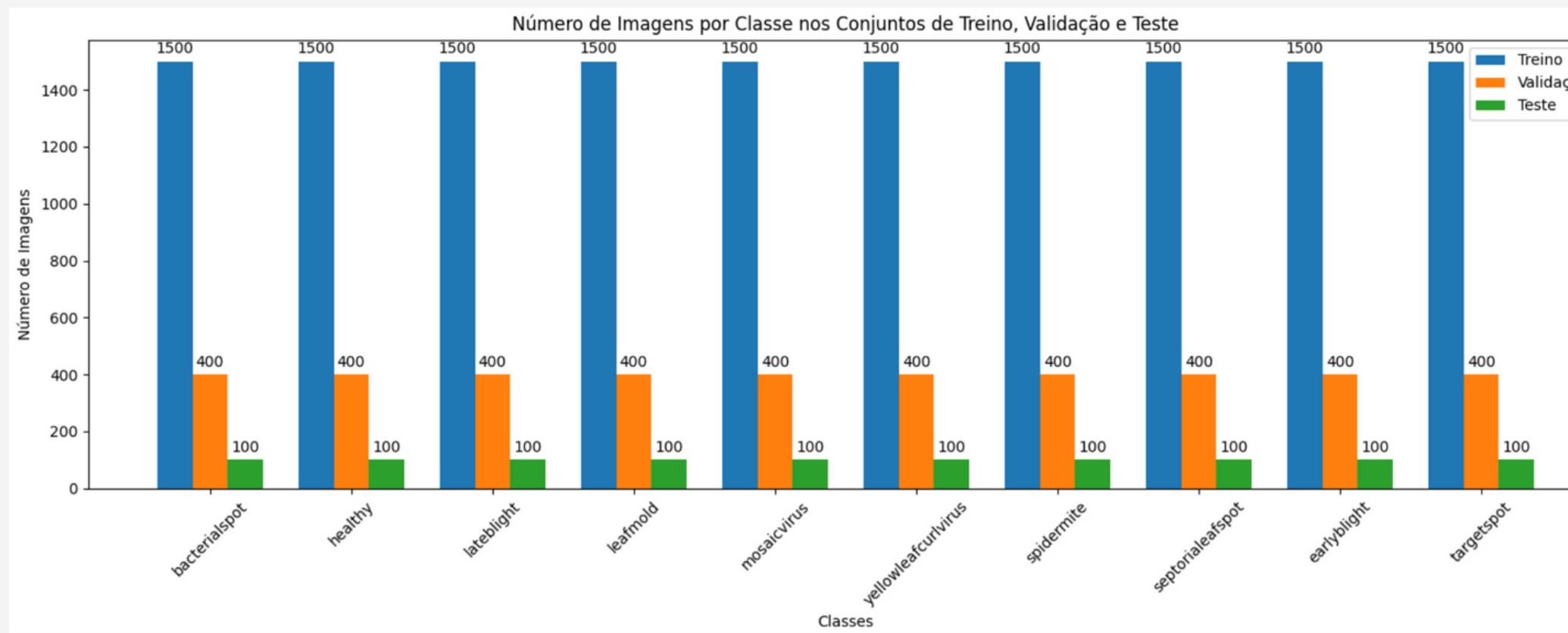
# ANÁLISE DE RESULTADOS

- O PSO com VGG16 alcançou um melhor AUC médio quanto maior o número de iterações.
- É possível visualizar com garantias que o VGG-16 neste cenário apresenta valores de AUC superiores ao Xception (97% vs 94%).
- No Xception um número maior de iterações não correspondeu necessariamente a melhores resultados.
- Grid Search, percebe-se que um aumento no número de épocas melhorou o AUC, indicando que um treinamento mais longo foi efetivo para o modelo VGG16.
- Os resultados do Random Search apresentaram melhorias com um número maior de épocas, mas com um intervalo mais amplo de hiperparâmetros, demonstrando que soluções eficientes podem ser encontradas fora dos valores tipicamente sugeridos.

# TREINO DOS MELHORES MODELOS

Foi utilizado early stopping e model checkpoint.

Objetivo definido com uma determinada taxa de threshold e foi utilizado 100% dos conjuntos



```
from tensorflow.keras.layers import Dense, Dropout, Flatten, BatchNormalization

def create_cnn_model(dropout_rate, learning_rate):
    base_model = VGG16(weights='imagenet', include_top=False, input_shape=(64, 64, 3))

    for layer in base_model.layers:
        layer.trainable = False

    # Tornar as últimas n camadas treináveis
    n_trainable_layers = 3
    for layer in base_model.layers[-n_trainable_layers:]:
        layer.trainable = True

    x = base_model.output
    x = Flatten()(x)
    x = Dense(128, activation='relu')(x)
    x = BatchNormalization()(x)
    x = Dropout(dropout_rate)(x)
    predictions = Dense(len(categories), activation='softmax')(x) # Output layer

    model = Model(inputs=base_model.input, outputs=predictions)

    model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=learning_rate),
                  loss='sparse_categorical_crossentropy',
                  metrics=['accuracy'])

    return model
```

# TREINO DOS MELHORES MÓDULO

```
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
import matplotlib.pyplot as plt
from tensorflow.keras.models import load_model

best_learning_rate = 0.001
dropout_rate = 0.225

val_accuracy_threshold = 0.965
early_stopping = EarlyStopping(monitor='val_loss', patience=20, min_delta=0.001, mode='min', verbose=1)
model_checkpoint = ModelCheckpoint('best_model.h5', monitor='val_accuracy', mode='max', save_best_only=True, verbose=0)

while True:
    print("A treinar o modelo...")
    best_model = create_cnn_model(dropout_rate, best_learning_rate)
    best_model_history = best_model.fit(x_train, y_train, epochs=60, batch_size=32, verbose=1, validation_data=(x_validation, y_validation), callbacks=[early_stopping, model_checkpoint])

    best_train_accuracy = best_model_history.history['accuracy'][-1]
    best_validation_accuracy = max(best_model_history.history['val_accuracy'])

    print("Melhor Accuracy de Treino: ", best_train_accuracy)
    print("Melhor Accuracy de Validação: ", best_validation_accuracy)

    if best_validation_accuracy >= val_accuracy_threshold:
        print("Atingiu a taxa de acerto desejada no conjunto de validação.")
        break
```

05

# RESULTADOS DOS MELHORES MODELOS

8 CLASSES

1 fase **91.38%**

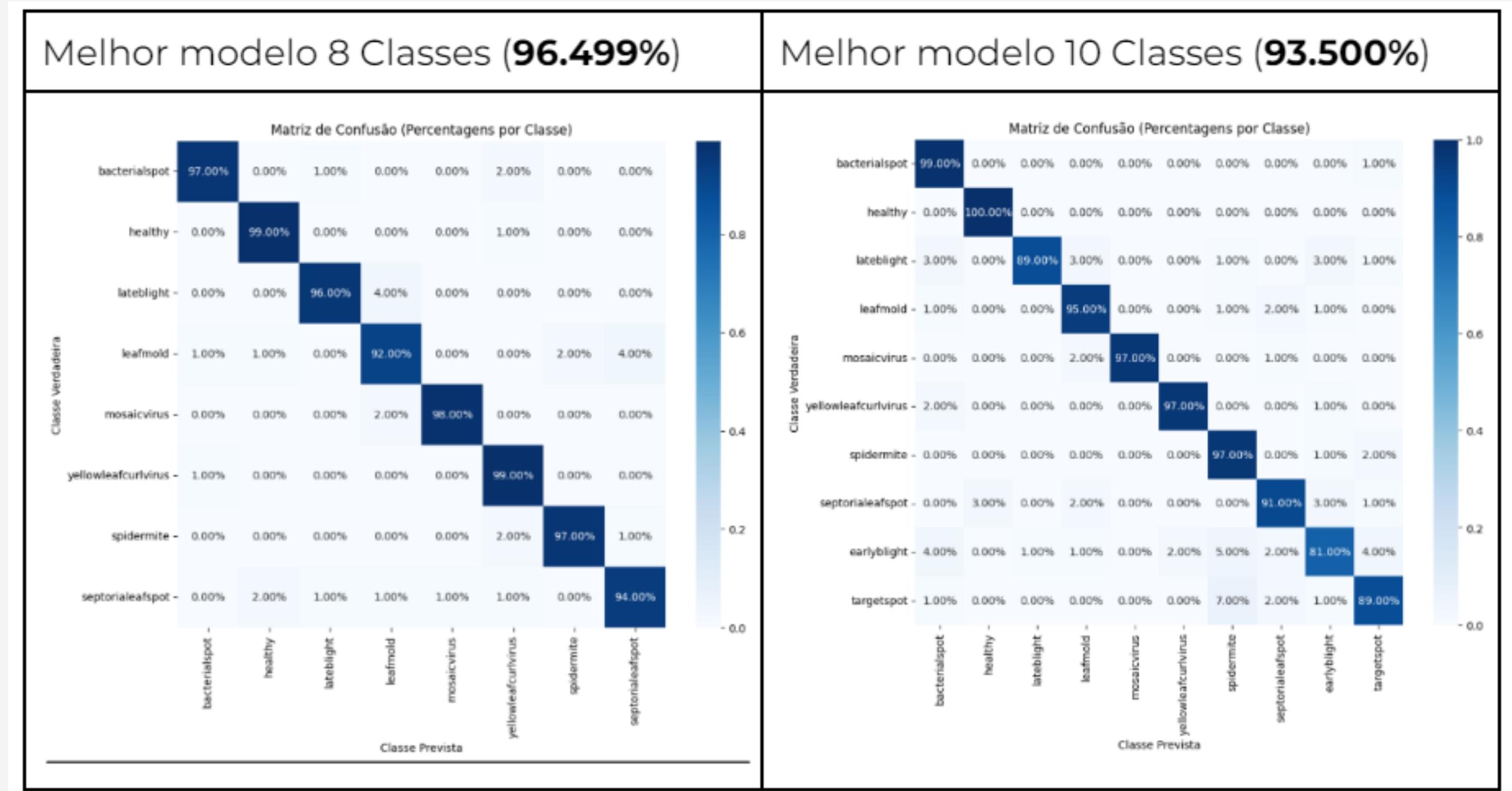
3 fase **96.499%**

10 CLASSES

1 fase **84.7%**

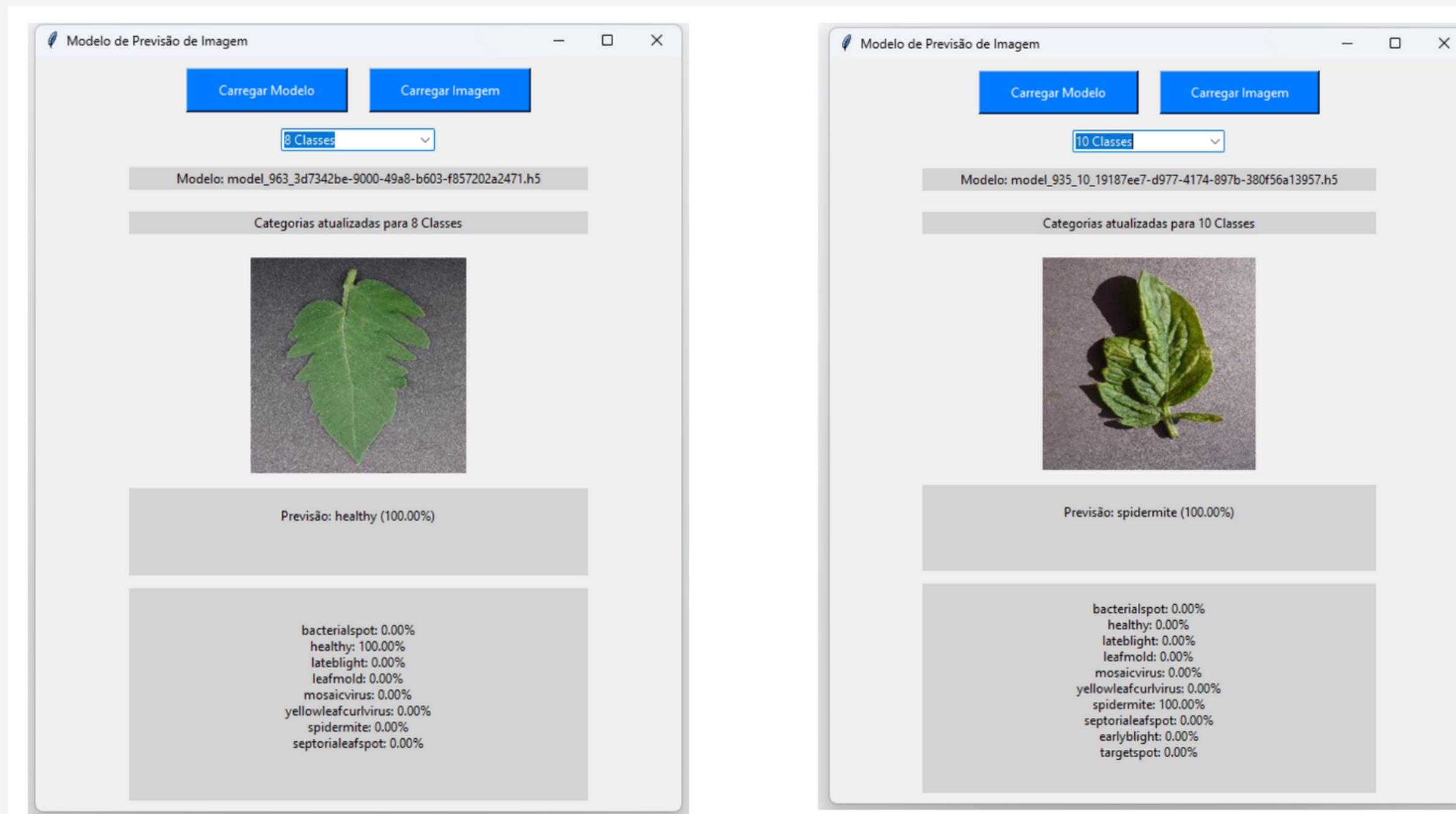
3 fase **93.5%**

# RESULTADOS DOS MELHORES MODELOS



06

# APLICAÇÃO REAL DO MODELO



07

# Q/A