MEIW – Mestrado em Engenharia Informática e Tecnologia Web

22292 Deep Learning Aplicado

Classificação de plantas medicinais (DIMPSAR: Dataset for Indian medicinal plant species analysis and recognition)

Ano letivo 2023/2024

DOCENTES:

António Cunha

Paulo Pombinho

Pedro Mestre

DISCENTE:

André Pereira [2302569]

Índice

Introdução

Este projeto e presente relatório estão inseridos no âmbito da Unidade Curricular de Deep Learning Aplicado, sendo esta presente no plano de estudos do Mestrado em Engenharia Informática e Tecnologia Web, curso ministrado pela Universidade Aberta e pela Universidade de Trás-os-Montes e Alto Douro. O objetivo deste trabalho é desenvolver um projeto com Redes neurais convolucionais (CNN's) , estudar as arquiteturas CNN's, os seus processos de treino e saber aplicá-las a problemas de visão por computador. No caso deste projeto em específico, pretende-se detetar e classificar diferentes tipos de plantas e folhas medicinais com recurso a CNN's.

Problema e Objetivo

A Ayurveda é um dos sistemas medicinais mais antigos praticados na Índia há milhares de anos, este visa tratar diversas doenças com custos mais baixos e efeitos colaterais indesejáveis em comparação com a medicina alopática. Cada parte das plantas medicinais, como raiz, folha, caule, fruto e semente, é composta por propriedades medicinais. O principal objetivo de criar um conjunto de dados de imagens de espécies de plantas medicinais indianas é promover o uso de práticas medicinais "ayurvédicas" e disseminar o conhecimento relacionado a plantas medicinais comuns que estão presentes ao nosso redor. Essa conscientização visa incentivar pesquisadores, educadores e praticantes no campo das plantas medicinais a enfrentar novos desafios. Além disso, isso levaria a um aumento na colheita de plantas medicinais, promovendo as melhores práticas de saúde entre a população e reduzindo os riscos relacionados à saúde. Especificamente, a criação dos conjuntos de dados pode proporcionar benefícios como identificação de espécies de plantas em aprendizado de máquina/aprendizado profundo, conservação da biodiversidade, pesquisa de plantas medicinais , análise fitoquímica, medicina "ayurvédica", educação e divulgação, conservação e uso sustentável. Em suma, a criação de um conjunto de dados de plantas medicinais indianas pode contribuir para o desenvolvimento de abordagens inovadoras e sustentáveis para sua utilização e conservação.

Dataset

As imagens de plantas medicinais capturadas em dispositivos móveis desempenham um papel significativo na identificação de espécies de plantas para pesquisas científicas. O conjunto de dados DIMPSAR (Dataset for Indian medicinal plant species analysis and recognition) contribui de maneira crucial para essa pesquisa, fornecendo dois conjuntos de dados que foram coletados: o conjunto de dados de folhas medicinais, composto por 80 espécies e 6900 imagens, capturadas sob várias condições, e o conjunto de dados de plantas medicinais, com 40 espécies e 5900 imagens, adquiridas sob condições diversas usando diferentes dispositivos móveis. incluindo características como múltiplas resoluções, variações de iluminação, fundos diversos e amostras capturadas em condições de tempo e estações do ano distintas. Destaca-se pela sua exclusividade, uma vez que não há um conjunto de dados padrão de órgãos de plantas para plantas medicinais indianas na literatura. A aquisição de imagens é desvinculada de fatores restritivos, permitindo a captura de imagens invariáveis em relação a estações, horários, condições de iluminação e fundo. Ao contrário de conjuntos de dados autoconstruídos mencionados na literatura, o DIMPSAR é único e abrange diversas condições de aquisição, como imagens ocultas, internas, externas, de sombra, e imagens com similaridade interclasse.

Tipo	Número de espécies	Total de imagens	Tipo de fundo
Dataset de folhas medicinais	80	6900	Plano não variável
Dataset de plantas medicinais	40	5900	Fundo variável

Tabela 1 - Descrição do Dataset

Estado da Arte

A pesquisa recente na área de classificação de espécies de plantas Ayurvédicas na Índia tem sido focada no desenvolvimento de abordagens inovadoras. O estudo conduzido por Pushpa e Rani (2023) destaca a criação do Ayur-PlantNet, uma rede neural convolucional (CNN) leve e imparcial para a classificação de quarenta espécies de plantas Ayurvédicas. Este trabalho aborda desafios comuns em classificação baseada em fotos, como má iluminação, folhas sobrepostas, oclusões de elementos de cena de domínios cruzados, semelhanças entre classes, variações intraclasse e elementos de cena desfocados. Ao comparar o Ayur-PlantNet com modelos pré-treinados, como Resnet34, Resnet50, VGG16, MobileNetV3_Large, EfficientNetwork_B4 e Densenet121, destaca-se que o Ayur-PlantNet atinge uma precisão de 92,27% com menos parâmetros treináveis e complexidade computacional reduzida em relação aos modelos pré-treinados, sendo a segunda rede mais precisa a MobileNetV3_Large.

A arquitetura do Ayur-PlantNet é considerada dominante em relação a outros modelos de aprendizado profundo, conforme evidenciado pelos resultados experimentais. O estudo enfatiza a eficácia do Ayur-PlantNet na superação dos desafios específicos associados à classificação de plantas Ayurvédicas, reforçando sua relevância no avanço da visão computacional aplicada à botânica medicinal.

O estudo conduzido por Roopashree et al. (2022) destaca o uso eficiente de técnicas de aprendizagem máquina para autenticação de ervas terapêuticas A pesquisa recente na interseção da IoT e reconhecimento de plantas medicinais usando descritores locais tem proporcionado avanços significativos. O emprego de descritores locais, como a técnica modificada de padrão binário local (LBP), revelou-se uma abordagem eficaz para medir características em imagens de folhas, resultando em uma precisão média de 96,22% em um conjunto de dados personalizado. Ao examinar os resultados obtidos por Roopashree et al., observamos a aplicação prática de descritores locais na identificação de características únicas de plantas medicinais.

Metodologia e Implementação

A abordagem adotada para a deteção e classificação de plantas medicinais envolve uma seleção de técnicas e ferramentas, para tentar atingir uma precisão robusta e confiável. A escolha da arquitetura pré-treinada desempenha um papel crucial na eficácia do modelo, e, para isso, optei por utilizar a MobileNetV3Large. A escolha da MobileNetV3Large baseia-se em sua capacidade de extrair características significativas de imagens, tornando-a uma escolha adequada para tarefas de classificação complexas, como a identificação de plantas medicinais. A sua arquitetura eficiente e leve permite o processamento rápido, sendo particularmente vantajosa para aplicações em dispositivos móveis e ambientes com recursos computacionais limitados, como é o caso.

Além da escolha da arquitetura, a metodologia abrange a organização e divisão cuidadosa dos conjuntos de dados de folhas e plantas, permitindo uma avaliação robusta do desempenho do modelo. A estratégia de treino, validação e teste é essencial para garantir a generalização do modelo em diferentes cenários.

A implementação do código é detalhada em seções distintas, cada uma desempenhando um papel específico, desde a importação de bibliotecas até a avaliação final do modelo. Inseri comentários no código para percebermos claramente o propósito e funcionamento de cada etapa. O uso de técnicas como data augmentation enriquece o conjunto de treino, melhorando a capacidade do modelo de generalizar para dados não vistos. As métricas de desempenho, como precisão, recall e matriz de confusão, são necessárias para avaliar a eficácia do modelo em cada conjunto de dados. Vamos então analisar as secções mais relevantes detalhadamente.

Preparação e Divisão do Conjunto de Dados das Plantas Medicinais

Após a importação de bibliotecas e preparação do ambiente, onde foi efetuado o download do dataset e o seu unzip, pretende-se preparar e dividir os conjuntos de dados, centrando-se na divisão das imagens em folhas e plantas medicinais. É começado por definir os diretórios principais para as folhas e plantas. Em seguida, cada subdiretório é percorrido e são coletadas informações sobre as imagens, como caminhos completos e classes correspondentes.

Começa-se por centrar o foco nas imagens de folhas medicinais. O diretório é explorado, e os dados de cada imagem são armazenados na lista rows_leaf. Esta lista é então convertida no DataFrame df_leaf. Em seguida, o conjunto de dados é dividido em treino e teste usando a função train_test_split, garantindo uma distribuição equitativa das classes. Exatamente a mesma lógica é aplicada para as imagens de plantas.

```
# Carregar conjunto de dados de folhas
main_folder_leaf = '/content/Indian Medicinal Leaves Image Datasets/Medicinal Leaf dataset'
sub_folders_leaf = Path(main_folder_leaf).glob('*')
rows_leaf = []
for folder_leaf in sub_folders_leaf:
    class_name_leaf = str(folder_leaf).replace(main_folder_leaf + "/", "")
    for file_leaf in os.listdir(folder_leaf):
        data_leaf = [str(folder_leaf) + "/" + file_leaf, class_name_leaf]
        rows_leaf.append(data_leaf)

# Armazenar todos os dados em um DataFrame pandas
df_leaf = pd.DataFrame(rows_leaf, columns=["Full Path", "Class Name"])
# Dividir conjunto de dados de folhas em treino e teste
x_train_leaf, x_test_leaf, y_train_leaf, y_test_leaf = train_test_split(
    df_leaf["Full Path"], df_leaf["Class Name"], test_size=0.2,
    random_state=7, shuffle=True, stratify=df_leaf["Class Name"]
)
# Armazenar as porções divididas em DataFrames
dftrain_leaf = pd.DataFrame({'Full Path': x_train_leaf, 'Class Name': y_train_leaf})
dftest_leaf = pd.DataFrame({'Full Path': x_test_leaf, 'Class Name': y_test_leaf})

# Imprimir DataFrames de treino e teste para folhas
print("DataFrame de Treino (Folhas):")
print(dftrain_leaf)
print("\nDataFrame de Teste (Folhas):")
print(dftest_leaf)

# Carregar conjunto de dados de plantas
main_folder_plant = '/content/Indian Medicinal Leaves Image Datasets/Medicinal plant dataset'
sub_folders_plant = Path(main_folder_plant).glob('*')
```

Figura 1 - Preparação e divisão do conjunto das Folhas

Funções para a análise e visualização de dados

Antes de se prosseguir com o modelo, trata-se de definir funções para a análise e visualização de dados, onde é pretendido definir 3 funções essenciais no contexto do modelo de classificação de plantas medicinais. A função generate_confusion_matrix cria e mostra a matriz de confusão com base nos resultados da predição do modelo. A função show_images mostra aleatoriamente imagens de um diretório especificado. A função predict_image realiza a predição de uma única imagem usando o modelo treinado, mostra a imagem original, a classe que foi predita e a probabilidade associada à predição.

```
# Função para gerar matriz de confusão de parâmetros:
# results - resultado da predição do modelo e labels - lista de classes
def generate_confusion_matrix(results, labels):
    cmn = results
    plt.figure(figsize=(100, 100))
    sns.set(font_scale=1.4)
    sns.heatmap(cmn, annot=True, fmt='.2f', annot_kws={"size": 16}, xticklabels=labels, yticklabels=labels, cmap="Greens")
    plt.show()

# Função para mostrar imagens de um diretório
def show_images(img_folder, num_rows=5, num_cols=2):
  fig=plt.figure(figsize=(20,20))
  classes = sorted(Path(img_folder).glob('*'))
  total_subplots = num_rows * num_cols
  i=0
  for cl in classes:
      files = os.listdir(cl)
      file = random.choice(files)
      image_path= os.path.join(cl, file)
      img=mpimg.imread(image_path)
      i=i+1
      fig.add_subplot(num_rows, num_cols, i)
      plt.imshow(img)
      if i == total_subplots:
          break
  plt.show()

# Função para fazer a predição de uma imagem de parâmetros:
# model - modelo treinado ou carregado de um arquivo; classes - lista de classes
# img_path - caminho para a imagem; # size - tamanho da imagem para redimensionar para o input da CNN
def predict_image(model, classes, img_path, size):
    # carregar a imagem
    image_orig = cv2.imread(img_path)
    image_orig = cv2.cvtColor(image_orig, cv2.COLOR_BGR2RGB)
    print(img_path)

    # pré-processar a imagem para classificação
    image = cv2.resize(image_orig, (size, size))
    image = img_to_array(image)
    image = np.expand_dims(image, axis=0)

    result = model.predict(image)
    class_id = result[0].argmax()
    classification = "Classe: {} ({:.2f}%)".format(classes[class_id], result[0][class_id] * 100)

    plt.figure(figsize=(8, 8))
    plt.imshow(image_orig)
    plt.show()
    print("[PREDIÇÃO] {}".format(classification))
```

Figura 2 - Funções para a análise e visualização de dados

Visualização de Amostras de Imagens

Nesta pequena secção apenas se vai utilizar a função show_images para visualizar aleatoriamente amostras de imagens dos conjuntos de dados.

```
show_images('/content/Indian Medicinal Leaves Image Datasets/Medicinal Leaf dataset')
```

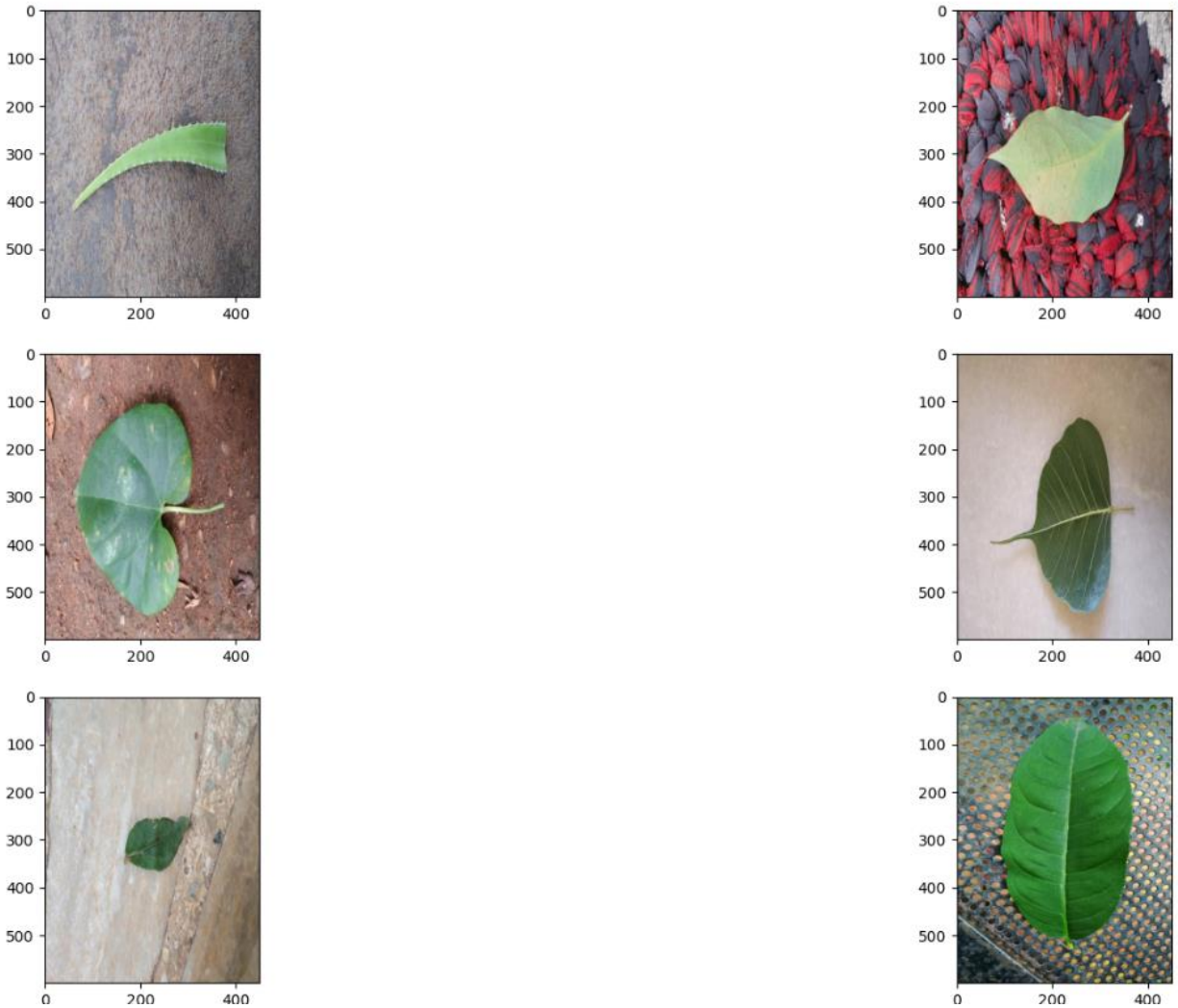


Figura 3 - Visualização do conjunto das folhas

Configuração dos hiperparâmetros e variáveis

Nesta seção, são definidas variáveis e hiperparâmetros para o treino do modelo. O tamanho da imagem de entrada (IMG_SIZE) é estabelecido como 224x224 px, com 3 canais de cor (RGB). Os hiperparâmetros de treino incluem 15 épocas, tamanho de lote de 32, taxa de aprendizagem de 0.0001 (uma taxa pequena garante uma convergência mais estável do modelo) e semente aleatória 42 (42 para garantir que os resultados se mantenham consistentes e possam ser repetidos de forma fácil). Também são definidos os números de classes para os conjuntos de dados de folhas (CLASSES_LEAF) e plantas (CLASSES_PLANT).

```
IMG_SIZE = 224
CHANNELS = 3

EPOCH = 15
BATCH_SIZE = 32
LR = 0.0001
SEED = 42

# função len(df_x["Class Name"].unique()) para obter o número de classes de cada conjunto.
# Pensei em definir números fixos, mas assim é uma forma mais dinâmica.
CLASSES_LEAF = len(df_leaf["Class Name"].unique())
CLASSES_PLANT = len(df_plant["Class Name"].unique())
```

Figura 4 - Configuração dos hiperparâmetros e variáveis

Criação dos modelos CNN para plantas e folhas

Na fase de construção dos modelos, após análise decidi por como base dos 2 modelos, a arquitetura pré-treinada MobileNetV3Large. Esta é uma arquitetura CNN projetada para tarefas de visão computacional, é notável pelo seu desempenho eficiente em termos de computação e tamanho do modelo. O modelo é escalável, podendo ser adaptado para diferentes tamanhos de entrada e requisitos de computação. Depois da base, eliminamos a camada de classificação e incorporamos camadas personalizadas, incluindo Global Average Pooling, camadas densas e uma camada de classificação softmax para a saída. A função compila os modelos utilizando a função de perda "categorical_crossentropy" e o otimizador Adam, especificando uma taxa de aprendizagem (LR). A camada de saída do MobileNetV3Large é removida e substituída por camadas adicionais personalizadas para se adequar à tarefa específica de classificação de folhas e plantas medicinais.

```
# Função para criar o modelo CNN
def create_cnn_model(classes):
    # Define o modelo base como MobileNetV3
    base_model = MobileNetV3Large(include_top=False, input_shape=(IMG_SIZE, IMG_SIZE, CHANNELS))

    # Adiciona camadas adicionais e camada de classificação ao modelo base
    x = GlobalAveragePooling2D()(base_model.output)
    x = Dense(512, activation='relu')(x)
    x = Dense(1024, activation='relu')(x)
    classifier = Dense(classes, activation='softmax')(x)

    # Define o modelo final
    model = Model(name="MedNet", inputs=base_model.input, outputs=classifier)

    # Compila o modelo
    opt = Adam(learning_rate=LR)
    model.compile(loss="categorical_crossentropy", optimizer=opt, metrics=['accuracy'])

    return model

# Criação do modelo para folhas
model_leaf = create_cnn_model(CLASSES_LEAF)
model_leaf.summary()

# Criação do modelo para plantas
model_plant = create_cnn_model(CLASSES_PLANT)
model_plant.summary()
```

Figura 5 - Construção dos modelos CNN para plantas e folhas

Preparação dos modelos CNN para folhas e plantas

Na etapa de preparação dos conjuntos de dados, recorre-se à biblioteca ImageDataGenerator do TensorFlow. Esta ferramenta ajuda a enriquecer a diversidade do conjunto de dados, aplicando transformações em tempo real durante o treino (destacam-se rotação, zoom, e inversão horizontal) contribuindo para um treino mais robusto e resistente a variações nas imagens.

A divisão do conjunto de dados em conjuntos de treino e validação é realizada, colocando 25% das amostras para validação. Esta prática permite avaliar o desempenho do modelo em dados não previamente observados. Para cada tarefa específica de classificação, seja para folhas ou plantas, são configurados geradores de dados distintos. Estes geradores incluem informações, como o caminho das imagens, as colunas dos rótulos, o tamanho do lote, a semente para reprodutibilidade, o modo de classe categórica e o tamanho alvo das imagens. Esta abordagem procura adequar o treino às características específicas de cada conjunto de dados, otimizando assim o desempenho dos modelos CNN para as tarefas de reconhecimento de folhas e plantas medicinais.

```
# Definir o gerador de dados de imagem para treino
train_datagen_leaf = ImageDataGenerator(validation_split=0.25)
train_generator_leaf = train_datagen_leaf.flow_from_dataframe(
    dataframe=dftrain_leaf,
    x_col="Full Path",
    y_col="Class Name",
    subset="training",
    batch_size=BATCH_SIZE,
    seed=SEED,
    shuffle=True,
    class_mode="categorical",
    target_size=(IMG_SIZE, IMG_SIZE)
)

# Definir o gerador de dados de imagem para validação
validation_generator_leaf = train_datagen_leaf.flow_from_dataframe(
    dataframe=dftrain_leaf,
    x_col="Full Path",
    y_col="Class Name",
    subset="validation",
    batch_size=BATCH_SIZE,
    seed=SEED,
    shuffle=True,
    class_mode="categorical",
    target_size=(IMG_SIZE, IMG_SIZE)
)

# Definir o gerador de dados de imagem para teste
test_datagen_leaf = ImageDataGenerator()
test_generator_leaf = test_datagen_leaf.flow_from_dataframe(
    dataframe=dftest_leaf,
    x_col="Full Path",
    y_col="Class Name",
    batch_size=BATCH_SIZE,
    seed=SEED,
    shuffle=False,
    class_mode="categorical",
    target_size=(IMG_SIZE, IMG_SIZE)
)
```

Figura 6 - Preparação dos modelos

Treino dos Modelos CNN

Nesta parte são treinados os dois modelos : um para folhas (model_leaf) e outro para plantas (model_plant). Cada modelo é treinado em conjuntos de dados específicos. O treino ocorre ao longo de um número definido de épocas (EPOCH = 15), ajustando os pesos do modelo para melhorar o desempenho. A validação é feita em um conjunto separado para observar o ajuste do modelo e evitar overfitting. Os resultados do treino são registados nos históricos (history_leaf e history_plant).

```
# Treinar o modelo para folhas
history_leaf = model_leaf.fit(
    train_generator_leaf,
    epochs=EPOCH,
    validation_data=validation_generator_leaf,
    batch_size=BATCH_SIZE
)

# Treinar o modelo para plantas
history_plant = model_plant.fit(
    train_generator_plant,
    epochs=EPOCH,
    validation_data=validation_generator_plant,
    batch_size=BATCH_SIZE
)
```

Figura 7 - Treino dos Modelos

Análise gráfica do desempenho dos modelos CNN

Depois de executados os treinos, para uma melhor compreensão dos mesmos , são exibidos os resultados do treino de modelos CNN para as duas categorias distintas: folhas e plantas. Os gráficos gerados mostram a evolução da precisão e da perda ao longo das diferentes épocas de treino. No caso das folhas, o primeiro gráfico retrata a precisão tanto no treino como na validação, enquanto o segundo gráfico ilustra as respetivas perdas. Para as plantas, os gráficos três e quatro representam, respetivamente, a precisão e a perda.

```python
# Fazer gráficos para os resultados de treino das folhas
acc_leaf = history_leaf.history['accuracy']
val_acc_leaf = history_leaf.history['val_accuracy']

loss_leaf = history_leaf.history['loss']
val_loss_leaf = history_leaf.history['val_loss']

plt.figure(figsize=(10, 10))
plt.subplot(2, 1, 1)
plt.plot(acc_leaf, label='Precisão de Treino (Folhas)')
plt.plot(val_acc_leaf, label='Precisão de Validação (Folhas)')
plt.legend(loc='lower right')
plt.ylabel('Precisão')
plt.ylim([min(plt.ylim()), 1])
plt.xlabel('Época')
plt.title('Precisão de Treino e Validação (Folhas)')

plt.subplot(2, 1, 2)
plt.plot(loss_leaf, label='Perda de Treino (Folhas)')
plt.plot(val_loss_leaf, label='Perda de Validação (Folhas)')
plt.legend(loc='upper right')
plt.ylabel('Perda')
plt.ylim([0, 1.0])
plt.title('Perda de Treino e Validação (Folhas)')
plt.xlabel('Época')
plt.show()
```

Figura 8 - Análise gráfica dos resultados de treino do modelo das folhas

Guardar os modelos de folhas e plantas

De seguida são guardados os modelos treinados para as categorias de folhas e plantas. Os modelos são armazenados em ficheiros com a extensão ".h5". Esta prática é fundamental para preservar os modelos após o treino, possibilitando a sua reutilização. Os nomes dos ficheiros são exibidos como confirmação que o processo de guardar os mesmos correu bem.

```
nome_do_modelo_leaf = "CNN-MedNet-Leaf.h5"
nome_do_modelo_plant = "CNN-MedNet-Plant.h5"

# Guardar os modelos
model_leaf.save(nome_do_modelo_leaf)
model_plant.save(nome_do_modelo_plant)

# Só para verificar que foi guardado
print(f"Modelo para folhas {nome_do_modelo_leaf} guardado com sucesso!")
print(f"Modelo para plantas {nome_do_modelo_plant} guardado com sucesso! ")
```

Figura 9 - Processo de guardar os modelos

Avaliação de Modelos e Desempenho

Nesta secção, carregamos e avaliamos os modelos previamente treinados para a classificação de folhas e plantas medicinais. Os modelos são avaliados nos respetivos conjuntos de teste, e os resultados, incluindo perda (Loss) e precisão (Accuracy), são apresentados. Além disso, realizamos previsões nos conjuntos de teste e calculamos métricas de desempenho, como precisão, recall, pontuação F1 e geramos matrizes de confusão para uma análise detalhada do desempenho em cada classe. As matrizes de confusão, apresentadas graficamente, oferecem uma visão detalhada do desempenho em cada classe, proporcionando uma compreensão visual da capacidade dos modelos em classificar corretamente diferentes espécies de plantas.

```
model_leaf_dir = "/content/CNN-MedNet-Leaf.h5"
model_plant_dir = "/content/CNN-MedNet-Plant.h5"
model_leaf = load_model(model_leaf_dir)
model_plant = load_model(model_plant_dir)

# Avaliar os modelos nos conjuntos de teste
evaluation_results_leaf = model_leaf.evaluate(test_generator_leaf)
evaluation_results_plant = model_plant.evaluate(test_generator_plant)

print("\nResultados de Avaliação para Folhas:")
print("Loss:", evaluation_results_leaf[0])
print("Accuracy:", evaluation_results_leaf[1])

print("\nResultados de Avaliação para Plantas:")
print("Loss:", evaluation_results_plant[0])
print("Accuracy:", evaluation_results_plant[1])

# Fazer previsões nos conjuntos de teste para folhas e plantas
predictions_leaf = model_leaf.predict(test_generator_leaf)
predictions_plant = model_plant.predict(test_generator_plant)

# Obter as classes previstas para folhas e plantas
y_pred_leaf = np.argmax(predictions_leaf, axis=1)
y_pred_plant = np.argmax(predictions_plant, axis=1)
labels_leaf = dftrain_leaf["Class Name"].unique()
labels_plant = dftrain_plant["Class Name"].unique()

# Calcular métricas de desempenho para folhas
print("\nMétricas de Desempenho para Folhas:")
print("Accuracy: ", accuracy_score(test_generator_leaf.labels, y_pred_leaf))
print("Precision: ", precision_score(test_generator_leaf.labels, y_pred_leaf, average="macro"))
print("Recall: ", recall_score(test_generator_leaf.labels, y_pred_leaf, average="macro"))
print("F1 Score: ", f1_score(test_generator_leaf.labels, y_pred_leaf, average="macro"))
matrix_leaf = confusion_matrix(test_generator_leaf.labels, y_pred_leaf)
generate_confusion_matrix(matrix_leaf, labels_leaf)

# Calcular métricas de desempenho para plantas
print("\nMétricas de Desempenho para Plantas:")
print("Accuracy: ", accuracy_score(test_generator_plant.labels, y_pred_plant))
print("Precision: ", precision_score(test_generator_plant.labels, y_pred_plant, average="macro"))
print("Recall: ", recall_score(test_generator_plant.labels, y_pred_plant, average="macro"))
print("F1 Score: ", f1_score(test_generator_plant.labels, y_pred_plant, average="macro"))
matrix_plant = confusion_matrix(test_generator_plant.labels, y_pred_plant)
generate_confusion_matrix(matrix_plant, labels_plant)
```

Figura 10 - Avaliação dos modelos e desempenho

Criar Predições

Nesta secção vemos a aplicação prática dos modelos treinados para a classificação de folhas e plantas medicinais. Para demonstrar a eficácia do modelo de folhas, é selecionada aleatoriamente uma pasta e um ficheiro dentro dessa pasta. A imagem correspondente é então utilizada para realizar uma previsão através da função predict_image. Este processo é igual para o modelo de plantas, onde novamente uma pasta e um ficheiro são escolhidos aleatoriamente para efetuar a previsão. As previsões resultantes são apresentadas com a classe identificada e a probabilidade associada, proporcionando uma visão prática da capacidade dos modelos em identificar e classificar imagens de folhas e plantas medicinais.

```
# Caminhos para os modelos das folhas e plantas
model_leaf_path = "/content/CNN-MedNet-Leaf.h5"
model_plant_path = "/content/CNN-MedNet-Plant.h5"

# Escolher aleatoriamente uma pasta e um ficheiro dentro dessa pasta para folhas
folder_leaf = random.choice(labels_leaf)
path_leaf = os.path.join(main_folder_leaf, folder_leaf)
file_leaf = random.choice(os.listdir(path_leaf))
img_path_leaf = os.path.join(path_leaf, file_leaf)

# Fazer a previsão para folhas
predict_image(load_model(model_leaf_path), labels_leaf, img_path_leaf, IMG_SIZE)

# Escolher aleatoriamente uma pasta e um ficheiro dentro dessa pasta para plantas
folder_plant = random.choice(labels_plant)
path_plant = os.path.join(main_folder_plant, folder_plant)
file_plant = random.choice(os.listdir(path_plant))
img_path_plant = os.path.join(path_plant, file_plant)

# Fazer a previsão para plantas
predict_image(load_model(model_plant_path), labels_plant, img_path_plant, IMG_SIZE)
```

Figura 11 - Predições em imagens aleatórias

Análise e discussão dos resultados

Preparação e Divisão do Conjunto de Dados das Plantas Medicinais

O output desta secção apresenta para as folhas, DataFrames de treino com 5523 registos e teste com 1381 registos. Já para as plantas, os DataFrames de treino e teste contêm 4756 e 1189 registos, respetivamente.

```
DataFrame de Treinamento (Folhas):
                                              Full Path  Class Name
3101  /content/Indian Medicinal Leaves Image Dataset...       Tulsi
2145  /content/Indian Medicinal Leaves Image Dataset...  Astma_weed
3881  /content/Indian Medicinal Leaves Image Dataset...     Insulin
1695  /content/Indian Medicinal Leaves Image Dataset...       Padri
5586  /content/Indian Medicinal Leaves Image Dataset...    Ganigale
...                                                 ...         ...
5770  /content/Indian Medicinal Leaves Image Dataset...      Chilly
6401  /content/Indian Medicinal Leaves Image Dataset...  Seethapala
5608  /content/Indian Medicinal Leaves Image Dataset...    Ganigale
1536  /content/Indian Medicinal Leaves Image Dataset...   Nelavembu
1961  /content/Indian Medicinal Leaves Image Dataset...       Nooni

[5523 rows x 2 columns]

DataFrame de Teste (Folhas):
                                              Full Path  Class Name
6470  /content/Indian Medicinal Leaves Image Dataset...  Seethapala
3920  /content/Indian Medicinal Leaves Image Dataset...     Insulin
6671  /content/Indian Medicinal Leaves Image Dataset...         Pea
348   /content/Indian Medicinal Leaves Image Dataset...      Bhrami
2847  /content/Indian Medicinal Leaves Image Dataset...       Guava
...                                                 ...         ...
1454  /content/Indian Medicinal Leaves Image Dataset...      Ganike
...
4949  /content/Indian Medicinal Leaves Image Dataset...   Lemon_grass
4425  /content/Indian Medicinal Leaves Image Dataset...     Betel_Nut

[1189 rows x 2 columns]
```

Figura 12 - Output da secção

Criação dos modelos CNN para plantas e folhas

Após a criação do modelo, o output apresenta o download do arquivo de pesos pré-treinados para a arquitetura MobileNetV3Large, especificamente o modelo com camadas convolucionais e sem a camada densa no topo (sem top). O processo é exibido em relação ao tamanho do arquivo e concluído com sucesso. A arquitetura do modelo MedNet é então detalhada.

A camada de entrada ("input_1") tem dimensões de 224x224 pixels e 3 canais de cor (RGB). As camadas subsequentes incluem operações de rescalonamento, convolução, normalização por lotes, ativação ReLU, multiplicação, e outras. O número total de parâmetros do modelo é indicado, dividido entre parâmetros treináveis e não treináveis. Isso fornece uma visão geral do tamanho do modelo e da quantidade de informações que ele contém. Este modelo, baseado na arquitetura MobileNetV3Large, será usado como 'backbone' para a tarefa de classificação de plantas medicinais.

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/mobilenet_v3/weights_mobilenet_v3_large_224_1.0_float_no_top_v2.h5
12683000/12683000 [==============================] - 0s 0us/step
Model: "MedNet"
__________________________________________________________________________________________________
 Layer (type)                Output Shape              Param #   Connected to
==================================================================================================
 input_1 (InputLayer)        [(None, 224, 224, 3)]     0         []

 rescaling (Rescaling)       (None, 224, 224, 3)       0         ['input_1[0][0]']

 Conv (Conv2D)               (None, 112, 112, 16)      432       ['rescaling[0][0]']

 Conv/BatchNorm (BatchNorma  (None, 112, 112, 16)      64        ['Conv[0][0]']
 lization)

 tf.__operators__.add (TFOp  (None, 112, 112, 16)      0         ['Conv/BatchNorm[0][0]']
 Lambda)

 re_lu (ReLU)                (None, 112, 112, 16)      0         ['tf.__operators__.add[0][0]']

 tf.math.multiply (TFOpLamb  (None, 112, 112, 16)      0         ['re_lu[0][0]']
 da)

 multiply (Multiply)         (None, 112, 112, 16)      0         ['Conv/BatchNorm[0][0]',
                                                                  'tf.math.multiply[0][0]']
...
Total params: 4054696 (15.47 MB)
Trainable params: 4030296 (15.37 MB)
Non-trainable params: 24400 (95.31 KB)
```

Figura 13 - Sumário do modelo criado

Preparação dos modelos CNN para folhas e plantas

O conjunto de dados foi processado com sucesso, resultando num total de 4143 imagens de treino, 1380 imagens de validação e 1377 imagens de teste para folhas. Da mesma forma, foram obtidas 3567 imagens de treino, 1189 imagens de validação e 1189 imagens de teste para plantas. No entanto, foi identificado um aviso relacionado a 4 nomes de arquivo de imagem inválidos no caminho especificado, que foram ignorados durante o processamento.

```
Found 4140 validated image filenames belonging to 80 classes.
Found 1379 validated image filenames belonging to 80 classes.
Found 1381 validated image filenames belonging to 80 classes.
/usr/local/lib/python3.10/dist-packages/keras/src/preprocessing/image.py:1137: UserWarning: Found 4 invalid image filename(s) in x_col="Full Path". These filename(s) will be ignored.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/keras/src/preprocessing/image.py:1137: UserWarning: Found 4 invalid image filename(s) in x_col="Full Path". These filename(s) will be ignored.
  warnings.warn(
Found 3567 validated image filenames belonging to 40 classes.
Found 1189 validated image filenames belonging to 40 classes.
Found 1189 validated image filenames belonging to 40 classes.
```

Figura 14 - Output da divisão das imagens

Treino dos Modelos CNN

Os modelos CNN foram treinados com sucesso para folhas e plantas. No caso das folhas, o treino foi realizado ao longo de 15 épocas, resultando numa precisão de validação de aproximadamente 90.79%. Durante as épocas, observou-se uma redução consistente na função de perda, indicando a melhoria do modelo. Para as plantas, o treino também ocorreu ao longo de 15 épocas, alcançando uma precisão de validação de cerca de 96.47%. Assim como no caso das folhas, houve uma diminuição constante na função de perda ao longo das épocas, evidenciando o aprimoramento do modelo.

Os tempos de treino foram significativos, sendo aproximadamente 221 segundos por época para folhas e 18 segundos por época para plantas. Estes resultados indicam um desempenho positivo dos modelos na classificação de imagens de folhas e plantas.

```
Epoch 1/15
130/130 [==============================] - 272s 2s/step - loss: 3.6197 - accuracy: 0.2186 - val_loss: 2.9009 - val_accuracy: 0.3125
Epoch 2/15
130/130 [==============================] - 227s 2s/step - loss: 1.1603 - accuracy: 0.7384 - val_loss: 2.1237 - val_accuracy: 0.4387
Epoch 3/15
130/130 [==============================] - 227s 2s/step - loss: 0.3160 - accuracy: 0.9355 - val_loss: 1.7411 - val_accuracy: 0.5410
Epoch 4/15
130/130 [==============================] - 224s 2s/step - loss: 0.1157 - accuracy: 0.9819 - val_loss: 1.3221 - val_accuracy: 0.6193
Epoch 5/15
130/130 [==============================] - 227s 2s/step - loss: 0.0533 - accuracy: 0.9957 - val_loss: 1.0233 - val_accuracy: 0.7128
Epoch 6/15
130/130 [==============================] - 246s 2s/step - loss: 0.0287 - accuracy: 0.9983 - val_loss: 0.9058 - val_accuracy: 0.7498
Epoch 7/15
130/130 [==============================] - 247s 2s/step - loss: 0.0180 - accuracy: 0.9988 - val_loss: 0.7463 - val_accuracy: 0.7766
Epoch 8/15
130/130 [==============================] - 219s 2s/step - loss: 0.0140 - accuracy: 0.9995 - val_loss: 0.5864 - val_accuracy: 0.8274
Epoch 9/15
130/130 [==============================] - 222s 2s/step - loss: 0.0175 - accuracy: 0.9973 - val_loss: 0.5728 - val_accuracy: 0.8405
Epoch 10/15
130/130 [==============================] - 214s 2s/step - loss: 0.0152 - accuracy: 0.9981 - val_loss: 0.4525 - val_accuracy: 0.8673
Epoch 11/15
130/130 [==============================] - 217s 2s/step - loss: 0.0162 - accuracy: 0.9969 - val_loss: 0.4822 - val_accuracy: 0.8651
Epoch 12/15
130/130 [==============================] - 220s 2s/step - loss: 0.0115 - accuracy: 0.9986 - val_loss: 0.3526 - val_accuracy: 0.8963
Epoch 13/15
...
Epoch 14/15
112/112 [==============================] - 17s 155ms/step - loss: 0.0030 - accuracy: 0.9997 - val_loss: 0.1363 - val_accuracy: 0.9664
Epoch 15/15
112/112 [==============================] - 18s 162ms/step - loss: 0.0015 - accuracy: 1.0000 - val_loss: 0.1302 - val_accuracy: 0.9647
```

Figura 15 - Resultado dos treinos

Análise gráfica do desempenho dos modelos CNN

Nesta secção são exibidos os resultados do treino de modelos CNN para as duas categorias distintas: folhas e plantas. Os gráficos gerados mostram a evolução da precisão e da perda ao longo das diferentes épocas de treino. Ao analisar os gráficos, é possível observar uma tendência positiva na precisão, indicando que os modelos estão a aprender de forma eficaz durante o treino. A proximidade entre as curvas de precisão de treino e validação sugere que os modelos não estão a sofrer overfitting, o que é positivo em termos de generalização. A redução contínua na perda, tanto no treino como na validação, sugere que os modelos estão a convergir e a melhorar as suas capacidades de classificação. Estes resultados indicam que os modelos estão a ser treinados eficientemente e estão prontos para realizar a classificação de novas imagens, distinguindo entre diferentes categorias de folhas e plantas.

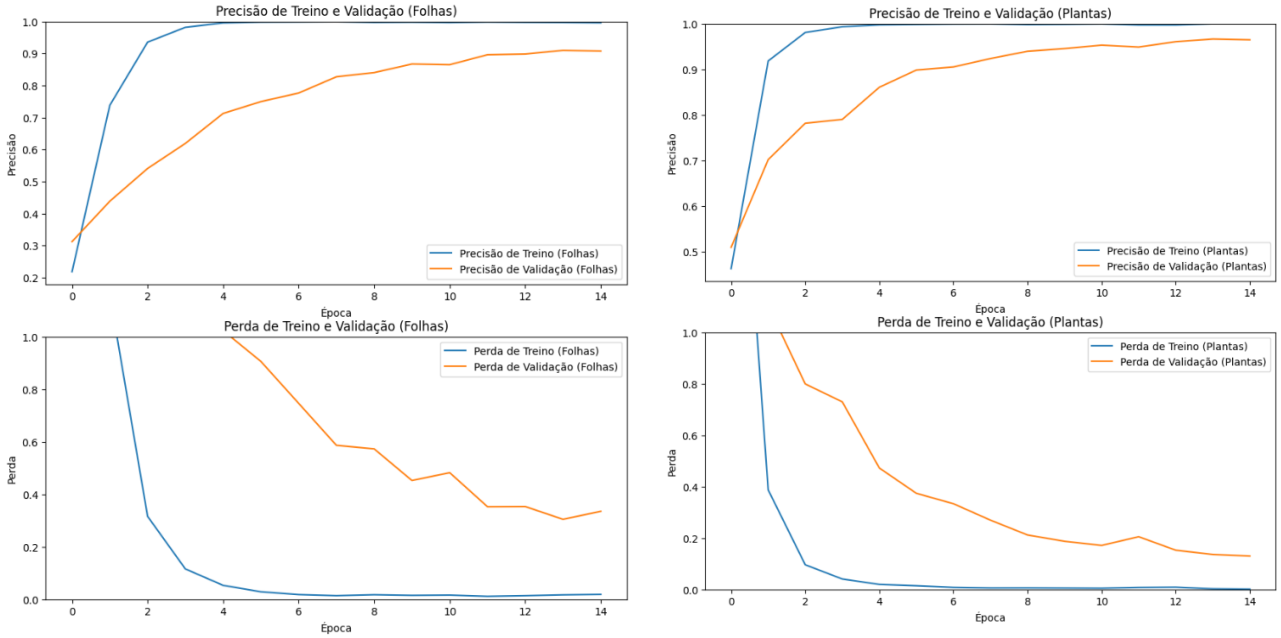


Figura 16 - Gráficos de precisão e perda dos modelos

Avaliação de Modelos e Desempenho

Durante a avaliação dos modelos treinados para folhas e plantas, observamos resultados promissores. Para o modelo de folhas, a perda (Loss) foi de 0.3280, enquanto a precisão (Accuracy) atingiu 91.21%. Já para o modelo de plantas, a perda foi significativamente menor, registando-se em 0.1449, e uma precisão notável de 96.13%. Ao analisarmos as métricas de desempenho para folhas, destacamos uma precisão geral de 93.03%, recall de 91.30%, e uma pontuação F1 de 91.40%. As métricas para plantas são ainda mais impressionantes, com uma precisão de 96.49%, recall de 96.07%, e pontuação F1 de 96.06%. As matrizes de confusão por serem 80x80 e 40x40 respetivamente, não irão ser incluídas no presente relatório pois a sua resolução seria muito baixa, podem, no entanto, ser consultadas no notebook do colab, que permite o zoom destas.

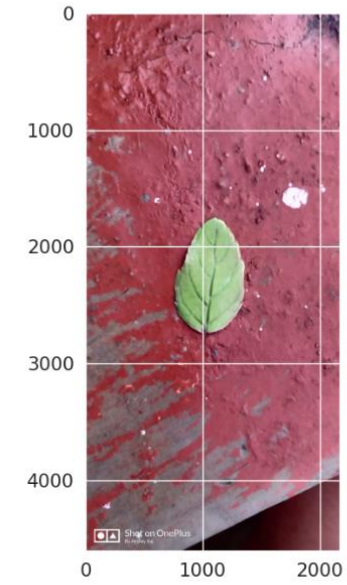
```
44/44 [==============================] - 69s 2s/step - loss: 0.3784 - accuracy: 0.9015
38/38 [==============================] - 7s 147ms/step - loss: 0.1645 - accuracy: 0.9588

Resultados de Avaliação para Folhas:
Loss: 0.37840139865875244
Accuracy: 0.9015206098556519

Resultados de Avaliação para Plantas:
Loss: 0.16450026631355286
Accuracy: 0.9587888717651367
44/44 [==============================] - 63s 1s/step
38/38 [==============================] - 4s 90ms/step

Métricas de Desempenho para Folhas:
Accuracy:  0.9015206372194062
Precision:  0.9201433034624852
Recall:  0.9049970628889141
F1 Score:  0.906118922262818
```

Figura 17 - Avaliação do modelo e métricas de desempenho

Criar Predições

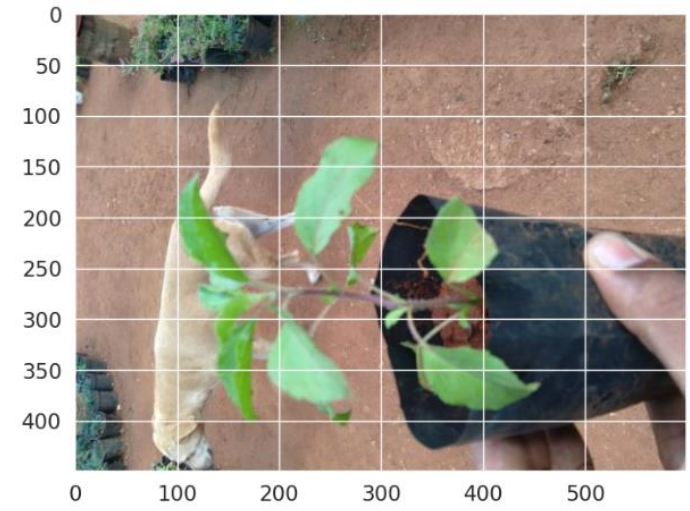
Nesta secção são escolhidas aleatoriamente imagens de uma planta e uma folha. A primeira imagem foi prevista como pertencendo à classe "Eucalyptus" com uma confiança de 91.23%. A segunda imagem foi classificada como pertencente à classe "Brahmi" com uma confiança de 100.00%. Ao analisar mais detalhadamente o output, observa-se que houve uma classificação incorreta para ambas as imagens, o que indica que o modelo pode apresentar desafios na correta identificação de certas classes, sugerindo a possibilidade de ajustes ou refinamentos no treino do modelo para melhorar a precisão, especialmente em casos de similaridade visual entre diferentes classes.

```
/content/Indian Medicinal Leaves Image Datasets/Medicinal Leaf dataset/Tulsi/IMG_20201003_180600.jpg
WARNING:tensorflow:6 out of the last 43 calls to <function Model.make_predict_function.<locals>.predic
1/1 [==============================] - 1s 1s/step
```



[PREDIÇÃO] Classe: Eucalyptus (91.23%)

```
/content/Indian Medicinal Leaves Image Datasets/Medicinal plant dataset/Tulasi/3559.jpg
1/1 [==============================] - 2s 2s/step
```



[PREDIÇÃO] Classe: Brahmi (100.00%)

Conclusões

Ao longo deste projeto, explorei a visão computacional aplicada à identificação e classificação de plantas medicinais através do conjunto de dados DIMPSAR. Utilizei redes neurais convolucionais (CNNs), especificamente a arquitetura MobileNetV3. A fase de pré-processamento dos dados envolveu a divisão adequada do conjunto de dados em treino e teste, permitindo uma avaliação robusta do desempenho dos modelos. As métricas de avaliação revelaram uma boa precisão.

Quanto à aplicação prática dos modelos em imagens aleatórias, ao analisar as predições específicas do modelo, as classificações foram incorretas. Essas discrepâncias nas predições podem ser atribuídas a diversos fatores:

- Similaridade Visual: a presença de características visuais semelhantes entre as diferentes espécies de plantas pode confundir o modelo, levando a previsões incorretas.
- Variações Intraclasse: diferenças consideráveis dentro da mesma classe podem resultar em dificuldades para o modelo distinguir entre subclasses, especialmente se o conjunto de dados não abranger adequadamente essas variações.
- Limitações do Conjunto de Dados: o conjunto de dados utilizado para treino pode não ser abrangente o suficiente e o modelo pode ter dificuldade em generalizar para todas as nuances das espécies de plantas .
- Melhorias na Arquitetura da Rede: Mesmo com uma arquitetura avançada como a MobileNetV3, sempre há espaço para melhorias e ajustes na arquitetura da rede neural para melhorar a capacidade de discriminação e até nos hiperpârametros.

Apesar dessas discrepâncias, o projeto como um todo destaca a eficácia da visão computacional na identificação de plantas medicinais, ressaltando a importância contínua do refinamento do modelo e expansão do conjunto de dados para aprimorar sua precisão e aplicabilidade prática. No entanto, é importante considerar a necessidade contínua de melhoria e expansão do conjunto de dados, visando melhorar ainda mais a robustez e generalização dos modelos desenvolvidos. Em última análise, este projeto destaca a importância que a visão computacional poderá vir a ter na área da botânica medicinal.

Bibliografia

- Pushpa, B. R., & Rani, N. S. (2023). DIMPSAR: Dataset for Indian medicinal plant species analysis and recognition. Data in Brief, 49, 109388.
- Pushpa, B. R., & Rani, N. S. (2023). Ayur-PlantNet: An unbiased light weight deep convolutional neural network for Indian Ayurvedic plant species classification. Journal of Applied Research on Medicinal and Aromatic Plants, 34, 100459.
- S. Roopashree, J. Anitha, T.R. Mahesh, V.V. Kumar, W. Viriyasitavat, A. Kaur, An IoT based authentication system for therapeutic herbs measured by local descriptors using machine learning approach, Measurement 200 (2022) 111484.
- Pushpa B R, Shobha Rani, “Indian medicinal leaves image datasets”, Mendeley Data V3 (2023), doi:10.17632/748f8jkphb.3.
- J.G. Thanikkal, A.K. Dubey, M.T. Thomas, An efficient mobile application for identification of immunity boosting medicinal plants using shape descriptor algorithm, Wirel. Pers. Commun. (2023) 1–17.
- J.G. Thanikkal, A.K. Dubey, M.T. Thomas, Deep-Morpho Algorithm (DMA) for medicinal leaves features extraction, Multimedia Tools Appl. (2023) 1–21.
- R.U. Rao, M.S. Lahari, K.P. Sri, K.Y. Srujana, D. Yaswanth, Identification of medicinal plants using deep learning, Int. J. Res. Appl. Sci. Eng. Technol. 10 (2022) 306–322.
- P.P. Kaur, S. Singh, Random forest classifier used for modelling and classification of herbal plants considering different features using machine learning, in: Mobile Radio Communications and 5G Networks: Proceedings of Second MRCN 2021, Springer Nature Singapore, Singapore, 2022, pp. 83–94.
- Divyasree, G., & Sheelarani, M. (2022). Ayurvedic leaf identification using deep learning model: VGG16. Available at SSRN 4091254.