Inteligência Computacional - 2023/2024

Fase 1 - Desenvolvimento sustentável

Identificação da saúde de produtos vegetais

2021110042 a2021110042@isec.pt | Jorge Ricardo Marques Duarte

Índice

Índice 1

Cap. 1: Descrição do caso de estudo e objetivos do problema 2

Cap. 2: Descrição da implementação dos algoritmos 3

Cap. 3: Análise de resultados 4

- Análise de Otimizadores com 10 classes 4
- Análise de funções de perda com 10 classes 5
- Análise de neurónios com 10 classes 5
- Análise de funções de ativação com 10 classes 6
- Análise de epochs de ativação com 10 classes 7
- Melhores Resultados (10 e 8 classes) 8

Cap. 4: Conclusões 9

Apresentação dos gráficos do melhor modelo 10

Teste e uso do modelo 10

Referências 10

Cap. 1: Descrição do caso de estudo e objetivos do problema

À medida que o planeta enfrenta desafios crescentes relacionados à segurança alimentar, a identificação precoce e eficaz de doenças em plantas e frutas emerge como uma solução vital no âmbito do desenvolvimento sustentável.

Esta capacidade de detetar e responder a doenças neste alimentos não apenas promove práticas agrícolas mais eficientes, mas também desempenha um papel fundamental na preservação da biodiversidade, na redução do uso de produtos químicos e na promoção de sistemas alimentares sustentáveis, dado que estes são os meus objetivos fundamentais para a escolha deste tema.

Durante a pesquisa para este trabalho, encontrei um conjunto de dados que se adequa perfeitamente ao âmbito do projeto. Trata-se do 'New Plant Diseases Dataset', um conjunto de dados que inclui 87.000 imagens que podem ser utilizadas e categorizadas.
Deste dataset enorme, acabei por escolher apenas analisar a fruta tomate de forma a obter um modelo mais preciso, onde foram utilizadas 20.000 imagens, 1500 por classe para treinar o modelo, 400 por classe para validar, e 100 por classe para testar o modelo.

As classes utilizadas para treinar este modelo foram as seguintes:

- Bacterialspot
- Healthy
- Lateblight
- Leafmold
- Mosaicvirus
- Yellowleafcurlvirus
- Spidermite
- Septorialeafspot
- Earlyblight
- Targetspot

O objetivo na criação do modelo é obter a melhor taxa de acerto global sobre as imagens utilizadas para teste, ou seja, quanto mais próximo este modelo se aproximar do 100%, melhor a sua eficiência para resolver o caso de estudo.

Com isto, serão utilizados vários otimizadores, totais de epochs diferentes, diferentes testes com totais de neurônios e camadas e serão apresentados dados como matrizes de confusão, e apresentação de resultados para métricas como a “accuracy”, a sensibilidade, a especificidade, f-measure e AUC.

Link referente ao dataset utilizado:
https://www.kaggle.com/datasets/vipoooool/new-plant-diseases-dataset/data

Cap. 2: Descrição da implementação dos algoritmos

No ficheiro de treino(train.ipynb) criei e treinei um modelo utilizando o TensorFlow/Keras e, em seguida, o modelo é avaliado com base em métricas como matriz de confusão, precisão por classe, precisão global, f-measure, auc, sensibilidade e guardo o modelo.

Vou descrever a implementação do código em etapas:

“Import” das bibliotecas:

- Começei por importar as bibliotecas como: TensorFlow/Keras, NumPy, PIL, Scikit-Learn, Seaborn e Matplotlib.

Definição dos diretórios de treino e categorias:

- Estou a especificar os diretórios de treino, validação e teste, bem como as classes de classificação.

Função load_images_from_folder:

- Esta função permite carregar as imagens dos diretórios e normalizá-las. A função cria duas listas: uma para as imagens (images) e outra para os rótulos (labels).
- As imagens são redimensionadas para o tamanho desejado, neste caso o tamanho utilizado foi (64x64) e normalizadas dividindo os valores dos pixels por 255.0.

Carregamento dos dados de treino e validação:

- Estou a definir os conjuntos de treino e validação usando a função load_images_from_folder que armazena as imagens em x_train e x_validation e os rótulos em y_train e y_validation.

Configurações do modelo:

- Aqui, podemos configurar os hiperparâmetros do modelo, como número de épocas, taxa de aprendizagem, otimizador a ser utilizado, função de perda, tamanho das imagens e número de neurônios em cada camada.

Criação do modelo MLP:

- De seguida, é criado um modelo sequencial de camadas. O modelo começa com uma camada “Flatten” para converter as imagens em vetores unidimensionais.
- Em seguida, é adicionado camadas densas com ativação ReLU, ou outro tipo de ativações e a camada de saída com ativação Softmax ou outro tipo que podemos especificar para classificar as 8 categorias.

Treino:

- Nesta fase, é utilizado o conjunto de treino e validação para treinar o modelo.

Avaliação do modelo:

- As previsões são feitas com o conjunto de teste e é calculada a matriz de confusão para avaliar o desempenho do modelo, a precisão por classe e a precisão global, F-Measure, ROC-AUC, a sensibilidade por classe e por último guardo modelo.

Cap. 3: Análise de resultados

Análise de Otimizadores com 10 classes

Global Accuracy	Total de Épocas	**Optimizador**	Função de Perda	Camada 1	Camada 2	Camada 3	Tamanho Imagens
70.90	10	**adam**	sparse_categorical_crossentropy	128 relu	64 relu	10 softmax	64x64
61.30	10	**sgd**	sparse_categorical_crossentropy	128 relu	64 relu	10 softmax	64x64
46.00	10	**adadelta**	sparse_categorical_crossentropy	128 relu	64 relu	10 softmax	64x64
70.80	10	**nadam**	sparse_categorical_crossentropy	128 relu	64 relu	10 softmax	64x64
65.60	10	**adagrad**	sparse_categorical_crossentropy	128 relu	64 relu	10 softmax	64x64
67.40	10	**rmsprop**	sparse_categorical_crossentropy	128 relu	64 relu	10 softmax	64x64

Análise de funções de perda com 10 classes

Global Accuracy	Total de Épocas	Optimizador	Função de Perda	Camada 1	Camada 2	Camada 3	Tamanho Imagens
70.90	10	adam	**sparse_categorical_crossentropy**	128 relu	64 relu	10 softmax	64x64
10.60	10	adam	**squared_hinge**	128 relu	64 relu	10 softmax	64x64

Análise de neurónios com 10 classes

Global Accuracy	Total de Épocas	Optimizador	Função de Perda	Camada 1	Camada 2	Camada 3	Camada 3/4	Tamanho Imagens
74.30	10	adam	sparse_categorical_crossentropy	**128** **relu**	**128** **relu**		10 softmax	64x64
70.90	10	adam	sparse_categorical_crossentropy	**128** **relu**	**64** **relu**		10 softmax	64x64
72.30	10	adam	sparse_categorical_crossentropy	**64** **relu**	**64** **relu**		10 softmax	64x64

63.80	10	adam	sparse_categorical_crossentropy	**256 relu**	**256 relu**		10 softmax	64x64
64.70	10	adam	sparse_categorical_crossentropy	**32 relu**	**64 relu**		10 softmax	64x64
67.00	10	adam	sparse_categorical_crossentropy	**64 relu**	**32 relu**		10 softmax	64x64
72.20	10	adam	sparse_categorical_crossentropy	**128 relu**	**128 relu**	**64 relu**	10 softmax	64x64
65.50	10	adam	sparse_categorical_crossentropy	**128 relu**	**64 relu**	**64 relu**	10 softmax	64x64
76.60	10	adam	sparse_categorical_crossentropy	**64 relu**	**64 relu**	**64 relu**	10 softmax	64x64

Análise de funções de ativação com 10 classes

76.60	10	adam	sparse_categorical_crossentropy	64 **relu**	64 **relu**	64 **relu**	10 softmax	64x64

80.00	10	adam	sparse_categorical_crossentropy	64 **elu**	64 **elu**	64 **elu**	10 softmax	64x64
28.80	10	adam	sparse_categorical_crossentropy	64 **sigmoid**	64 **sigmoid**	64 **sigmoid**	10 softmax	64x64
10.80	10	adam	sparse_categorical_crossentropy	64 **tanh**	64 **tanh**	64 **tanh**	10 softmax	64x64
70.00	10	adam	sparse_categorical_crossentropy	64 **leaky_relu**	64 **leaky_relu**	64 **leaky_relu**	10 softmax	64x64
74.60	10	adam	sparse_categorical_crossentropy	64 **swish**	64 **swish**	64 **swish**	10 softmax	64x64

Análise de epochs de ativação com 10 classes

80.00	10	adam	sparse_categorical_crossentropy	64 elu	64 elu	64 elu	10 softmax	64x64
83.00	50	adam	sparse_categorical_crossentropy	64 elu	64 elu	64 elu	10 softmax	64x64

83.90	100	adam	sparse_categorical_crossentropy	64 elu	64 elu	64 elu	10 softmax	64x64

Melhores Resultados (10 e 8 classes)

83.90	100	adam	sparse_categorical_crossentropy	64 elu	64 elu	64 elu	10 softmax	64x64
82.50	100	adam	sparse_categorical_crossentropy	128 elu	64 elu	64 elu	10 softmax	64x64
91.38	100	adam	sparse_categorical_crossentropy	64 elu	64 elu	64 elu	8 softmax	64x64
90.38	100	adam	sparse_categorical_crossentropy	64 elu	64 elu	32 elu	8 softmax	64x64
90.00	100	adam	sparse_categorical_crossentropy	128 elu	64 elu	32 elu	8 softmax	64x64

Cap. 4: Conclusões

Em um problema com 8 classes, as mesmas configurações obtiveram excelentes resultados com accuracy global de 91,38%, destacando a robustez do modelo. Variações dessas configurações, como a escolha do número de neurônios nas camadas ocultas, também foram testadas e alcançaram bons resultados. Em resumo, a escolha cuidadosa dos hiperparâmetros, incluindo o otimizador, a função de perda, o número de neurônios e a função de ativação, pode afetar significativamente o desempenho do modelo.

O conjunto de hiperparâmetros que levou ao melhor resultado (accuracy global de 83,90%) inclui o otimizador "Adam," uma função de perda "Sparse Categorical Cross Entropy," funções de ativação "ELU," uma única camada oculta com 64 neurônios, e imagens de tamanho 64x64.

Experimentar diferentes combinações de hiperparâmetros é essencial para encontrar a configuração ideal para o problema de classificação. Além disso, um treino mais longo, com um número maior de épocas, pode levar a um melhor desempenho, desde que o modelo não sofra de overfitting.

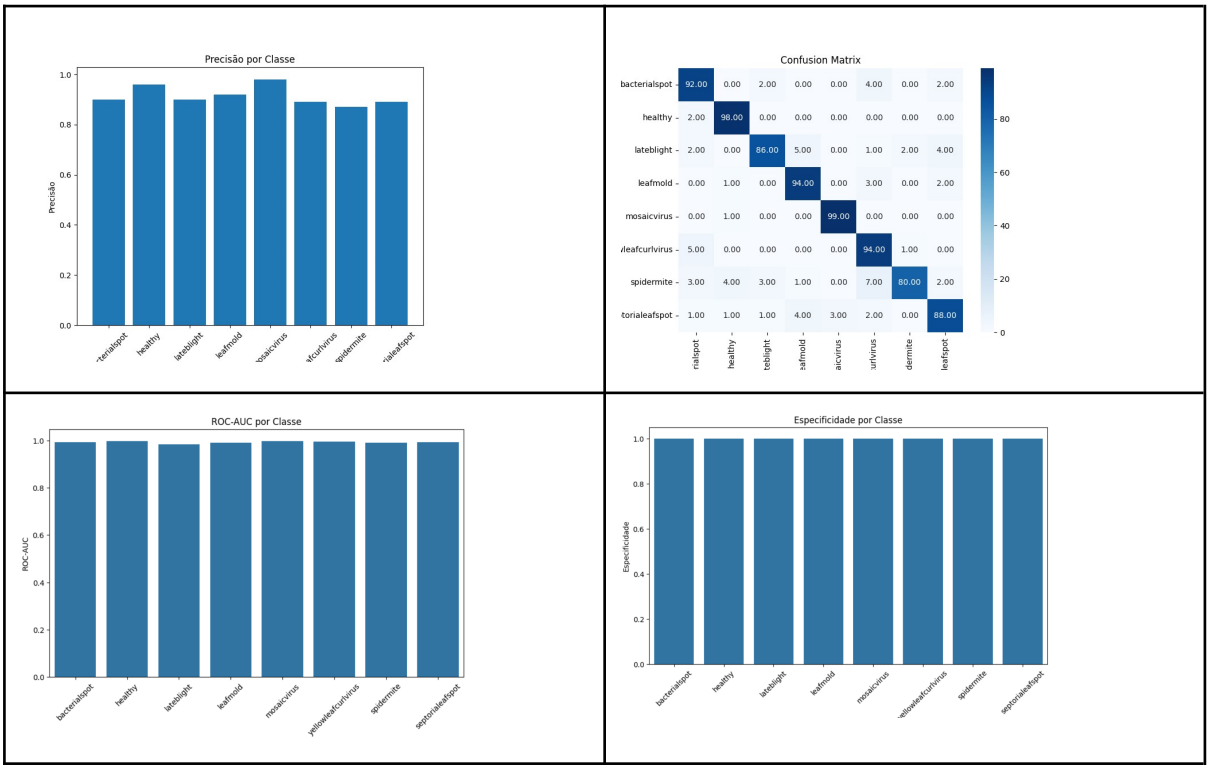
Observei que aumentar o número de épocas de treino de 10 para 50 e, em seguida, para 100, melhorou a accuracy global, sendo que um treino com 100 épocas (83,90%) superou 50 épocas (83,00%) e 10 épocas (80,00%) em termos de desempenho.

Obtive dificuldade em superar valores superiores a 85% com 10 classes, dado que o modelo tinha dificuldades em separar a classe earlyblight da lateblight. Isto deve-se ao facto que o modelo ficava muito em dúvida se era uma ou a outra dado a aproximação das duas. Com isto acabei por remover a earlyblight das classes ficando apenas com a lateblight.

Outra das categorias que o modelo obtinha muitas dificuldades a identificar com taxas percentuais de 80% era a targetspot, dado que acabei por decidir remover esta classe da análise, criando desta forma modelos com 8 classes mas com taxas de acerto global superiores a 90% facilmente.

Foi possível observar outros tipos de funções de perda como a binary_crossentropy, mas dado que só se aplicam a modelos que tenham apenas duas classes, acabei por não ter apresentado resultados porque não se aplica neste projeto, mas foram efetuados testes e análises de modo a perceber a sua utilização.
É possível notar que a redução da resolução das imagens de 256x256 para 64x64, acaba por ter impactos a nível da taxa de acerto global do modelo, dado que acabamos por perder alguns pormenores e detalhes da imagem.

Apresentação dos gráficos do melhor modelo



Teste e uso do modelo

Acabei no fim por criar um novo notebook em python chamado de “test-model.ipynb”, em que carrega o melhor modelo, é podemos definir a imagem a ser testada e após a execução, é apresentada o rótulo (label) da classe que o modelo acha ser a imagem, assim como a probabilidade por classe.

Predicted Label: healthy
Probability by class:
- bacterialspot: 0.02%
- healthy: 99.98%
- lateblight: 0.00%
- ....

Referências

- **Kaggle Dataset: New Plant Diseases Dataset** - Este é o link para o conjunto de dados fornecido no Kaggle, que serviu como base para o trabalho.
- **Documentação do TensorFlow e Keras** - Referenciou-se a documentação oficial do TensorFlow e Keras para obter informações sobre as funções, classes e métodos dessas bibliotecas de aprendizagem.
- Documentações de outras bibliotecas como Pillow, plotlib, Panda, etc.