PROJETO FASE III

Inteligência Computacional 2023/2024

Licenciatura em Engenharia Informática

Dinis Meireles de Sousa Falcão / a2020130403@isec.pt

Kevin Fernando Pereira Rodrigues / a2013010749@isec.pt

ÍNDICE

DESCRIÇÃO DO PROBLEMA

O objetivo desta fase do trabalho consiste na implementação e validação das metodologias de Aprendizagem automática, estudadas no âmbito da disciplina de **Inteligência Computacional**, a um caso de estudo real, analisado nas **Fase I** e **II**.

Após definir o modelo da uma rede neuronal para o problema (**Fase I**), estudo de um algoritmo swarm para otimização de híper-parâmetros (**Fase II**), segue-se a fase de construção do modelo final melhorado recorrendo a **transferência de aprendizagem de uma rede CNN** ou **de "Large Language Model"**, treinada com grandes conjuntos de dados - **imagens**, e posterior ajuste e otimização da arquitetura com base em algoritmos de enxame para pesquisa dos melhores híper-parâmetros (**Hidden Layer Size** e **Dropout**).

Foram executadas as seguintes tarefas:

I. Dividir dados em Treino/Validação e Teste;

II. Ajustar a dimensão e balancear o "*dataset*";

III. Selecuinar uma arquitetura pré-treinada (**VGG16**);

IV. Definir uma rede densa para as "*top layers*";

V. Otimização de Híper-Parâmetros da rede densa;

VI. Determinar melhor configuração de Híper-Parâmetros;

VII. Anotar os dados em Excel;

VIII. Análise do desempenho final.

DESCRIÇÃO DAS METODOLOGIAS UTILIZADAS

Em relação às metodologias utilizadas na realização da **Fase III do Projeto**, estas foram:

I. **VGG16:**
   a. **Visual Geometry Group 16**, é uma arquitetura de rede neural convolucional profundamente desenvolvida para tarefas de reconhecimento de imagem. Proposta pelo grupo Visual Geometry Group da Universidade de Oxford, a VGG16 é conhecida pela sua profundidade, consistindo em 16 camadas de operações convolucionais e de pooling. Esta destacou-se em competições de reconhecimento de imagem devido à sua capacidade de aprender representações complexas de características visuais hierárquicas.

II. **Algoritmo PSO:**
   a. O **Algoritmo Particule Swarm Otimization** é uma técnica de otimização inspirada no comportamento social de organismos coletivos. Neste algoritmo, cada "partícula" representa uma solução candidata num espaço de busca multidimensional. As partículas movem-se pelo espaço de busca com base em experiências próprias e nas melhores experiências globais do enxame. Esse processo iterativo visa encontrar a solução ótima ou aproximadamente ótima para um determinado problema de otimização.

III. **Validação Cruzada:**
   a. É uma técnica essencial na avaliação de desempenho de modelos de aprendizagem. Envolve a divisão do conjunto de dados em partes, chamadas de "*folds*". O modelo é treinado em alguns desses "*folds*" e

testado nos "*folds*" restantes. Este procedimento é repetido várias vezes, permitindo uma avaliação mais robusta do desempenho do modelo, reduzindo assim o impacto da variabilidade nos conjuntos de dados.

IV. Pesquisa em Grelha:

a. A **Grid Search** é uma estratégia sistemática de seleção de hiper-parâmetros em modelos de aprendizagem. Nesta abordagem, uma grelha de combinações de valores de hiper-parâmetros é especificada antecipadamente, e o desempenho do modelo é avaliado para cada combinação usando validação cruzada ou outro método de avaliação. A combinação que resulta no melhor desempenho é então escolhida como a configuração final do modelo.

V. Random Search:

a. A **Pesquisa Aleatória** é uma abordagem alternativa à **Pesquisa em Grelha** para otimização de hiper-parâmetros. Neste método, as combinações de hiperparâmetros não são pré-determinadas numa grelha fixa, mas são escolhidas aleatoriamente a partir de distribuições predefinidas. Isto permite explorar de maneira mais eficiente o espaço de hiper-parâmetros, especialmente quando a influência de determinados parâmetros é desconhecida. A **Random Search** pode ser mais eficaz em encontrar configurações de hiper-parâmetros que resultam num bom desempenho do modelo.

APRESENTAÇÃO DA ARQUITETURA DO CÓDIGO

DIAGRAMA DE COMPONENTES

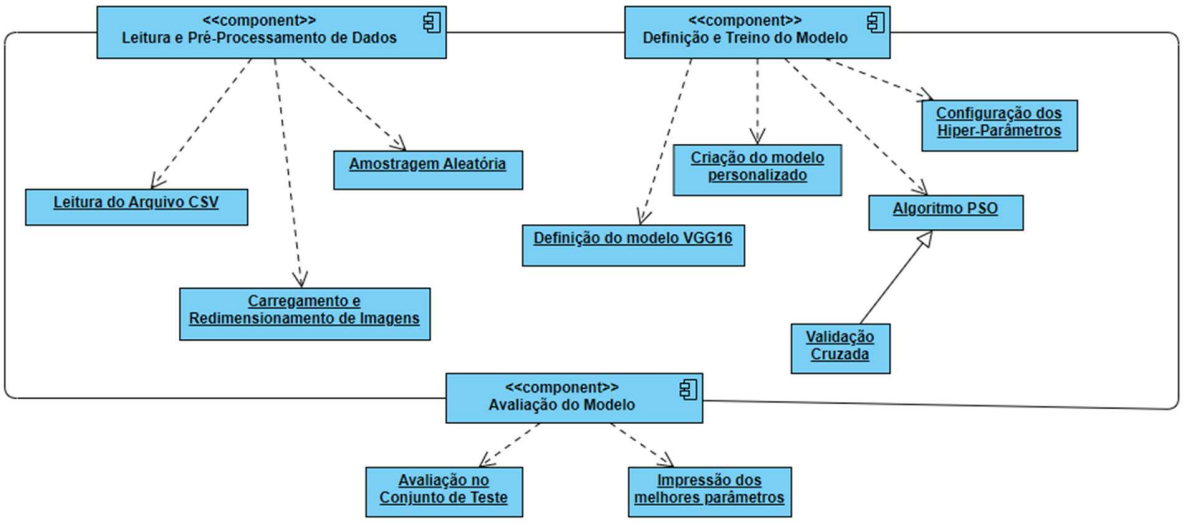
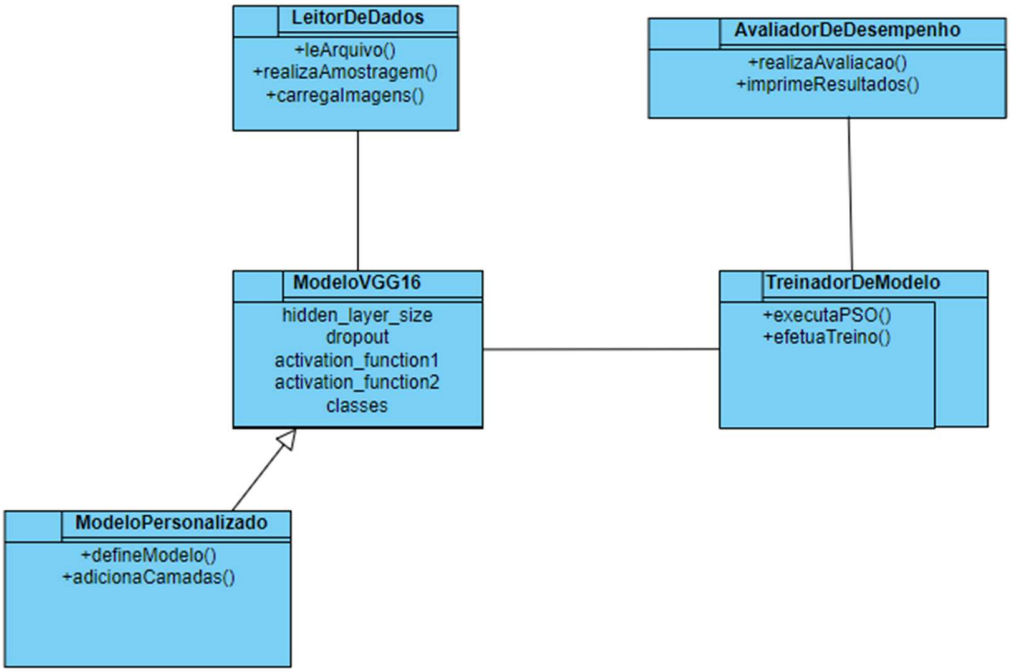


DIAGRAMA DE CLASSES



DESCRIÇÃO DA IMPLEMENTAÇÃO DOS ALGORITMOS

O código desenvolvido em "*faseIIIPSOcruzada.py*" implementa o algoritmo de otimização **PSO** para ajustar dois hiper-parâmetros de uma rede neuronal convolucional (**CNN**) baseada na arquitetura **VGG16**.

I. Função "*evaluate*":

- **a.** função objetivo que o **PSO** tenta minimizar. Neste caso, a função devolve o negativo da "*accuracy*" média nos dados de teste ao longo de várias "*folds*" de validação cruzada;
- **b.** cria e treina um modelo **VGG16** modificado com base nos parâmetros fornecidos pelo **PSO**;
- **c.** Utiliza a validação cruzada "***KFold***" para treinar e avaliar o modelo em diferentes sub-conjuntos de treino e validação.

II. Otimizador PSO:

- **a.** Utiliza a biblioteca "***SwarmPackagePy***" para criar uma instância do otimizador **PSO**;
- **b.** Define o número de partículas ("**n**"), a função objetivo ("**function**"), os limites inferiores ("**lb**") e superiores ("**ub**") para cada dimensão do espaço de busca, a dimensão do espaço de busca ("**dimension**") e o número de iterações ("**iteration**").

III. Execução do PSO e Impressão dos Resultados:

- **a.** Executa o otimizador **PSO** com a função "*pso*";
- **b.** Após a conclusão das iterações, imprime os melhores parâmetros encontrados pelo **PSO**.

O código desenvolvido em "*faseIIISearch.py*" implementa os algoritmos de **Grid Search** e **Random Search** para ajustar dois hiper-parâmetros de uma rede neuronal convolucional (**CNN**) baseada na arquitetura **VGG16**.

I. **Grid Search:**
   a. **Definição do Espaço de Parâmetros:**
      i. O código define um espaço de parâmetros a serem pesquisados. Estes incluem diferentes ***hidden_layer_sizes*** e ***dropout_rate***;
      ii. *"param_grid*" é um dicionário que contém as combinações de parâmetros a serem avaliadas na pesquisa em grelha.
   b. **Loop de Pesquisa em Grelha:**
      i. Utiliza dois loops para iterar todas as combinações de parâmetros definidas no espaço de parâmetros;
      ii. Cria um modelo utilizando a função "*create_model*" com os parâmetros atuais;
      iii. Treina o modelo no conjunto de treino em 10 épocas;
      iv. Avalia o modelo no conjunto de validação e regista a *accuracy*;
      v. Atualiza a melhor pontuação ("*best_grid_score*") e o melhor modelo ("*best_grid_model*") se a accuracy atual for maior do que a melhor pontuação registada.

II. **Random Search:**
   a. **Definição do Espaço de Parâmetros Aleatórios:**
      i. O código utiliza um espaço de parâmetros aleatórios que é semelhante ao utilizado na pesquisa em grelha;
      ii. "*n_iter_search*" define o número de iterações para a pesquisa aleatória.
   b. **Loop de Pesquisa Aleatória:**
      i. Itera "*n_iter_search*" vezes;

ii. Gera aleatoriamente um conjunto de parâmetros a partir do espaço de parâmetros aleatórios;
iii. Cria um modelo utilizando a função "*create_model*" com os parâmetros gerados aleatoriamente;
iv. Treina o modelo no conjunto de treino em 10 épocas;
v. Avalia o modelo no conjunto de validação e registra a accuracy;
vi. Atualiza a melhor pontuação ("*best_random_score*") e o melhor modelo ("*best_random_model*") se a accuracy atual for maior do que a melhor pontuação registada.

III. Avaliação Final no Conjunto de Teste:

a. Os melhores modelos encontrados pela pesquisa em grelha e pesquisa aleatória são avaliados no conjunto de teste;
b. As previsões são comparadas com as classes reais, e a accuracy é calculada utilizando a função "*accuracy_score*";
c. Os resultados de desempenho são então impressos para comparar as accuracy obtidas pelas duas abordagens de otimização.

ANÁLISE DOS RESULTADOS

Ao longo deste trabalho foram realizados vários testes e modificações ao longo da realização. Alguns dos resultados obtidos poderão ser vistos no ficheiro ***resultados.xlsx***, sendo que também vamos mostrar alguns dos mesmos em seguida:

I. **VGG16 (default):**
   a. **Número de camadas ocultas: 256**
   b. **Função de ativação: ReLu**
   c. **Taxa de Dropout: 0.5**
   d. **6 classes**
   e. **Função de ativação: SoftMax**
   f. **Resultados:**

```
Epoch 1/10
16/16 [==============================] - 99s 6s/step - loss: 8.3404 - accuracy: 0.6961 - val_loss: 7.2450 - val_accuracy: 0.8070
Epoch 2/10
16/16 [==============================] - 78s 5s/step - loss: 2.3816 - accuracy: 0.9000 - val_loss: 7.6796 - val_accuracy: 0.8187
Epoch 3/10
16/16 [==============================] - 82s 5s/step - loss: 1.0672 - accuracy: 0.9588 - val_loss: 5.7605 - val_accuracy: 0.8538
Epoch 4/10
16/16 [==============================] - 80s 5s/step - loss: 0.5571 - accuracy: 0.9745 - val_loss: 4.8587 - val_accuracy: 0.8596
Epoch 5/10
16/16 [==============================] - 80s 5s/step - loss: 0.3679 - accuracy: 0.9804 - val_loss: 5.1106 - val_accuracy: 0.8480
Epoch 6/10
16/16 [==============================] - 80s 5s/step - loss: 0.4828 - accuracy: 0.9804 - val_loss: 4.8109 - val_accuracy: 0.8538
Epoch 7/10
16/16 [==============================] - 81s 5s/step - loss: 0.2103 - accuracy: 0.9824 - val_loss: 5.1393 - val_accuracy: 0.8538
Epoch 8/10
16/16 [==============================] - 81s 5s/step - loss: 0.4005 - accuracy: 0.9804 - val_loss: 5.5030 - val_accuracy: 0.8538
Epoch 9/10
16/16 [==============================] - 82s 5s/step - loss: 0.2128 - accuracy: 0.9882 - val_loss: 5.1152 - val_accuracy: 0.8713
Epoch 10/10
16/16 [==============================] - 82s 5s/step - loss: 0.0637 - accuracy: 0.9922 - val_loss: 6.4190 - val_accuracy: 0.8538
6/6 [==============================] - 22s 3s/step - loss: 4.5437 - accuracy: 0.8772
Acuracy no conjunto de teste: 0.8771929740905762
```

Melhor Accuracy no Conjunto de treino: 0.9882 = 98.82%

Melhor Accuracy no Conjunto de Validação: 0.8713 = 87.13%

Accuracy no Conjunto de Teste: 0.8771 = 87.71%

II. **VGG16 (PSO com validação cruzada):**

a. **Número de camads ocultas: 165.35**

b. **Função de ativação: ReLu**

c. **Taxa de Dropout: 0.3**

d. **6 classes**

e. **Função de ativação: SoftMax**

f. **Resultados do melhor:**

```
Epoch 1/5
13/13 [==============================] - 66s 5s/step - loss: 0.2086 - accuracy: 0.9902 - val_loss: 7.5440e-06 - val_accuracy: 1.0000
Epoch 2/5
13/13 [==============================] - 63s 5s/step - loss: 0.1452 - accuracy: 0.9828 - val_loss: 3.9736e-08 - val_accuracy: 1.0000
Epoch 3/5
13/13 [==============================] - 63s 5s/step - loss: 0.1793 - accuracy: 0.9951 - val_loss: 5.8436e-09 - val_accuracy: 1.0000
Epoch 4/5
13/13 [==============================] - 63s 5s/step - loss: 0.0202 - accuracy: 0.9975 - val_loss: 1.4025e-08 - val_accuracy: 1.0000
Epoch 5/5
13/13 [==============================] - 62s 5s/step - loss: 0.1764 - accuracy: 0.9926 - val_loss: 1.8816e-07 - val_accuracy: 1.0000
6/6 [==============================] - 22s 3s/step - loss: 2.3834 - accuracy: 0.9240
```

Melhor Accuracy no Conjunto de Treino: 0.9975 = 99.75%

Melhor Accuracy no Conjunto de Validação: 1.0000 = 100%

Accuracy no Conjunto de Teste: 0.9240 = 92.40%

III. **VGG16 (Pesquisa em Grelha):**

a. **Número de camads ocultas: 256**

b. **Função de ativação: ReLu**

c. **Taxa de Dropout: 0.2**

d. **6 classes**

e. **Função de ativação: SoftMax**

f. **Resultados do melhor:**

```
6/6 [==============================] - 26s 3s/step - loss: 4.9081 - accuracy: 0.8480
Parameters: hidden_size=(128,), dropout_rate=0.2, Score: 0.847953200340271
6/6 [==============================] - 25s 3s/step - loss: 7.1000 - accuracy: 0.8596
Parameters: hidden_size=(128,), dropout_rate=0.5, Score: 0.859649121761322
6/6 [==============================] - 27s 3s/step - loss: 1.6624 - accuracy: 0.8538
Parameters: hidden_size=(128,), dropout_rate=0.8, Score: 0.8538011908531189
6/6 [==============================] - 24s 3s/step - loss: 2.3676 - accuracy: 0.9006
Parameters: hidden_size=(256,), dropout_rate=0.2, Score: 0.9005848169326782
6/6 [==============================] - 27s 4s/step - loss: 4.7280 - accuracy: 0.8772
Parameters: hidden_size=(256,), dropout_rate=0.5, Score: 0.8771929740905762
6/6 [==============================] - 26s 3s/step - loss: 2.2021 - accuracy: 0.8596
Parameters: hidden_size=(256,), dropout_rate=0.8, Score: 0.859649121761322
6/6 [==============================] - 25s 3s/step - loss: 6.3564 - accuracy: 0.8421
Parameters: hidden_size=(512,), dropout_rate=0.2, Score: 0.8421052694320679
6/6 [==============================] - 26s 3s/step - loss: 5.5015 - accuracy: 0.8421
Parameters: hidden_size=(512,), dropout_rate=0.5, Score: 0.8421052694320679
6/6 [==============================] - 23s 3s/step - loss: 2.1425 - accuracy: 0.8246
Parameters: hidden_size=(512,), dropout_rate=0.8, Score: 0.8245614171028137
```

Accuracy no conjunto de Teste: 0.9006 = 90.06%

Melhor Score: 0.9005848169326782

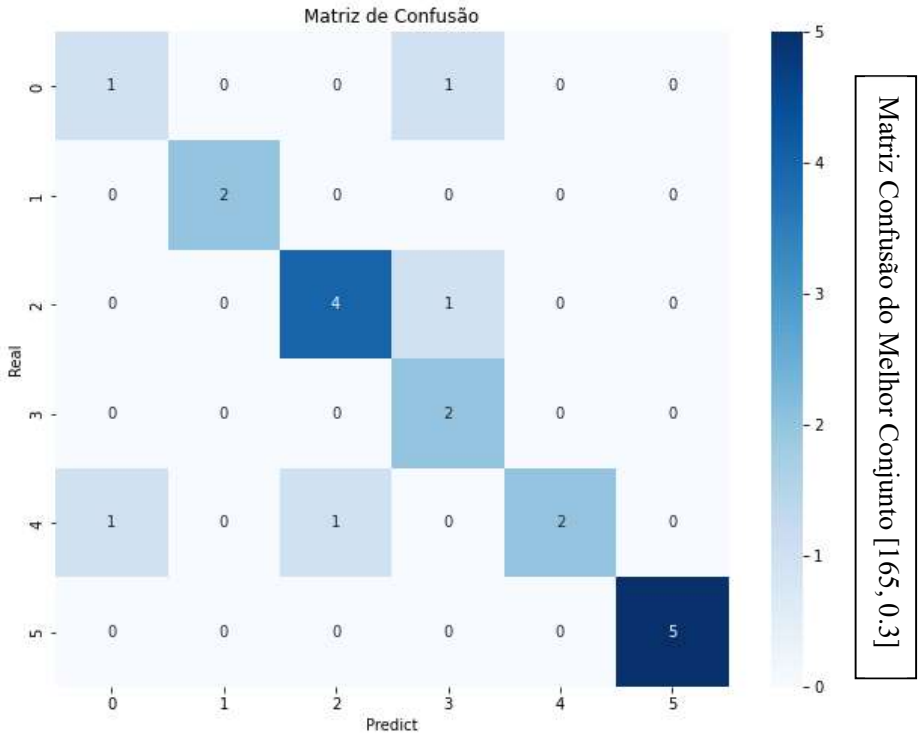
IV. VGG16 (Pesquisa Aleatória):

a. **Número de camads ocultas: 128**
b. **Função de ativação: ReLu**
c. **Taxa de Dropout: 0.8**
d. **6 classes**
e. **Função de ativação: SoftMax**
f. **Resultados do melhor:**

```
6/6 [==============================] - 23s 3s/step - loss: 4.2588 - accuracy: 0.8772
Parameters: hidden_size=(128,), dropout_rate=0.2, Score: 0.8771929740905762
6/6 [==============================] - 23s 3s/step - loss: 6.5690 - accuracy: 0.8538
Parameters: hidden_size=(128,), dropout_rate=0.2, Score: 0.8538011908531189
6/6 [==============================] - 23s 3s/step - loss: 3.2047 - accuracy: 0.8655
Parameters: hidden_size=(256,), dropout_rate=0.2, Score: 0.8654970526695251
6/6 [==============================] - 24s 3s/step - loss: 6.7250 - accuracy: 0.8772
Parameters: hidden_size=(128,), dropout_rate=0.2, Score: 0.8771929740905762
6/6 [==============================] - 23s 3s/step - loss: 2.6114 - accuracy: 0.9006
Parameters: hidden_size=(128,), dropout_rate=0.8, Score: 0.9005848169326782
6/6 [==============================] - 22s 3s/step - loss: 2.6516 - accuracy: 0.8480
Parameters: hidden_size=(512,), dropout_rate=0.8, Score: 0.847953200340271
6/6 [==============================] - 23s 3s/step - loss: 3.9254 - accuracy: 0.8596
Parameters: hidden_size=(128,), dropout_rate=0.5, Score: 0.859649121761322
6/6 [==============================] - 24s 3s/step - loss: 3.6444 - accuracy: 0.8830
Parameters: hidden_size=(512,), dropout_rate=0.2, Score: 0.8830409646034241
6/6 [==============================] - 27s 4s/step - loss: 8.0621 - accuracy: 0.8538
Parameters: hidden_size=(128,), dropout_rate=0.2, Score: 0.8538011908531189
6/6 [==============================] - 26s 4s/step - loss: 6.3942 - accuracy: 0.8480
Parameters: hidden_size=(256,), dropout_rate=0.5, Score: 0.847953200340271
6/6 [==============================] - 29s 4s/step
```

Accuracy no conjunto de Teste: 0.9006 = 90.06%

Melhor Score: 0.9005848169326782



Matriz Confusão do Melhor Conjunto [165, 0.3]

CONCLUSÕES

Relembrando a **Fase I do Projeto** de Inteligência Computacional, foram obtidos os seguintes resultados para a rede meuronal criada:

- **Número de neurónios da camada 1:** 300
- **Accuracy:** 40.00%

Na **Fase II do Projeto**, e com um conhecimento da matéra alargado relativamente à fase anterior, obtivemos uma melhoria dos resultados:

- **Número de neurónios da camada 1:** 147
- **Número de neurónios da camada 2:** 193
- **Accuracy:** 55.58%

Nesta última **Fase III do Projeto**, e com um conhecimento quase total de todas as técnicas abordadas nas aulas, melhorámos significativamente os resultados anteriores:

- **Número de neurónios da camada 1:** 165
- **Taxa de Dropout:** 0.3
- **Accuracy:** 92.40%

Sendo assim, e de acordo com os resultados obtidos, podemos concluir que houve uma melhoria progressive nos resultados ao longo das diferentes fases deste projeto: **aumento de cerca de 14% da Fase I para a Fase III**, e **aumento de cerca de 37% da Fase II para a Fase III**.

BIBLIOGRAFIA

I. https://chat.openai.com/

II. https://github.com/SISDevelop/SwarmPackagePy/tree/master

III. https://moodle.isec.pt/moodle/course/view.php?id=20497

IV. https://www.learnpython.org/

V. https://scikit-learn.org/stable/supervised_learning.html#supervised-learning

FIM