Instituto Superior
de Engenharia
Politécnico de Coimbra

Inteligência Computacional

Engenharia Informática 2023/2024

PROJECTO – FASE III

Alexandre Ferreira – 2021138219

Carlos Pinto – 2021155627

Descrição do Problema

Neste trabalho, foi abordada a aplicação de técnicas de Machine Learning (**ML**) na resolução de um problema de classificação de imagens. O problema escolhido foi a classificação de imagens em três categorias distintas: "**Fogo**," "**Fumo**," e "**Nada**." Este problema tem aplicações práticas em várias áreas, incluindo vigilância de incêndios, deteção de fumos e análise de imagens gerais.

O objetivo é avaliar a existência de focos de incendio e a sua classificação quanto a presença de fumo, chama ou nenhum destes com base em imagens.

Descrição das Metodologias utilizadas

O código utiliza uma abordagem de otimização de hiperparâmetros para treino de uma rede neuronal utilizando os algoritmos GSA (Gravitational Search Algorithm) e PSO (Particle Swarm Optimization).

GSA (Gravitational Search Algorithm):

O GSA é um algoritmo de otimização baseado na interação gravitacional entre corpos massivos no espaço. Nesse contexto, as soluções candidatas são tratadas como corpos com massa, e a atração gravitacional entre esses corpos é utilizada para explorar o espaço de pesquisa na procura melhores soluções.

No código, o GSA é aplicado para otimizar os hiperparâmetros da rede neuronal, como o número de épocas de treino e o número de neurónios nas camadas ocultas.

PSO (Particle Swarm Optimization):

O PSO é um algoritmo de otimização inspirado no comportamento de enxames na natureza, como bandos de pássaros ou cardumes de peixes. Cada solução candidata é representada por uma partícula que se move no espaço de pesquisa,

ajustando a sua posição com base na sua própria experiência e na experiência das partículas vizinhas.

No contexto do código, o PSO é usado para encontrar os melhores conjuntos de hiperparâmetros para otimizar o desempenho da rede neuronal.

Transfer Learning com Xception:

O código utiliza a arquitetura pré-treinada **Xception** como base para a construção da rede neuronal. A Xception é uma **rede neuronal convolucional** profunda que é eficaz para tarefas de classificação de imagens. Este modelo pré-treinado é adaptado para a tarefa específica em questão, ajustando os pesos durante o treino com o conjunto de dados fornecido.

Apresentação da Arquitetura de Código

A arquitetura do código resume-se a:

- **Importar os dados;**
- **Definição da Função de Treino e Avaliação;**
- **Escolha ou criação do modelo;**
    - **Criação da Base da Xception;**
    - **Adição de "top layers" ao Modelo;**
    - **Compilação e Treino do Modelo;**
- **Avaliação do Modelo;**
    - **Insere os resultados com os respetivos hiperparâmetros num ficheiro excel**
- **Aplicação do GSA e PSO para Otimização;**
    - **Guardar os melhores modelos;**
- **Avaliação e Métricas;**

Descrição da Implementação dos algoritmos

Importar os dados:

- Utilização da biblioteca Keras para carregar conjuntos de dados de treino, teste e validação como objetos image_dataset_from_directory.

```
ds_treino = image_dataset_from_directory(treino, batch_size=32, image_size=(larg,alt))
ds_teste = image_dataset_from_directory(teste, batch_size=32, image_size=(larg,alt))
ds_validacao = image_dataset_from_directory(validacao, batch_size=32, image_size=(larg, alt))
```

Definição da Função de Treino e Avaliação:

- Implementação da função train_and_evaluate que realiza o treino da rede e a avaliação do modelo.

```
def train_and_evaluate(params, x = 0):
    global excel
    nepochs = int(params[0])
    nneurons1 = int(params[1])
    nneurons2 = int(params[2])
```

Escolha ou criação do modelo:

- Decidir se deve ser importar um modelo treinado anteriormente ou criar um novo:

  Criação da Base da Xception:

  - Inicialização da base da Xception, uma rede neuronal convolucional pré-treinada para tarefas de classificação de imagens.

  Adição de Camadas ao Modelo:

  - Adição de camadas adicionais ao modelo sequencial para personalização da arquitetura.

  Compilação e Treino do Modelo:

  - Compilação do modelo com otimizador Adam, função de perda 'sparse_categorical_crossentropy', e métricas de accuracy.
  - Treino do modelo com os dados de treino (ds_treino) e validação (ds_validacao) especificados.

```
base_model = Xception(weights='imagenet')
            # Cria um modelo sequencial
            modelo = tf.keras.Sequential()
            # Adiciona a base da Xception ao modelo
            modelo.add(base_model
            modelo.add(tf.keras.layers.Flatten(input_shape=(larg, alt,3)))  #input
            modelo.add(tf.keras.layers.Dense(nneurons1, activation=atv))  #hidden
            modelo.add(tf.keras.layers.Dense(nneurons2, activation=atv))  #hidden
            modelo.add(tf.keras.layers.Dense(num_classes, activation='softmax'))  #output
            #TREINO
            modelo.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=
               ['accuracy'])
            #treina o modelo nepochs vezes
            modelo.fit(ds_treino, epochs=nepochs, batch_size=32, validation_data=ds_validacao)
```

Avaliação do Modelo:

- Realização da fase de teste utilizando o conjunto de dados de teste (ds_teste).

```
        y_pred = modelo.predict(x_test)
        y_pred_classes = np.argmax(y_pred, axis=1)
        accuracy = accuracy_score(y_test, y_pred_classes)
```

Inserção dos dados num ficheiro excel:

- A cada teste é adicionada uma linha no ficheiro output.xlsx, isto permite que caso o programa seja interrompido os resultados não sejam perdidos.

```
        excel = excel.append({'nepochs': nepochs, 'nneurons1': nneurons1, 'nneurons2':
nneurons2, 'accuracy': accuracy}, ignore_index=True)
```

Aplicação do GSA e PSO para Otimização:

- Utilização dos algoritmos GSA e PSO para otimizar os hiperparâmetros da rede neuronal.

```
alggsa  = spp.gsa(nparticles,train_and_evaluate,lb,ub,3,gsa_iter)
best_gsa = alggsa.get_Gbest()
alh  = spp.pso(nparticles,train_and_evaluate,lb,ub,3,gsa_iter)
best_pso = alh.get_Gbest()
```

Avaliação e Métricas:

- Cálculo de diversas métricas de desempenho após o treino e teste do modelo.

```
        #Resultados da previsão
        conf_matrix = confusion_matrix(y_test, y_pred_classes)
        result = classification_report(y_test, y_pred_classes, zero_division=1)
        aucovr = (roc_auc_score(y_test, y_pred, multi_class='ovr'))
        aucovo = (roc_auc_score(y_test, y_pred, multi_class='ovo'))

        sens, spec, _ , _ = precision_recall_fscore_support(y_test, y_pred_classes,
average='weighted', zero_division=1)
        fmeasure = 2*(sens * spec)/(sens + spec)
```

Análise de Resultados

Gsa Vs Pso:

Melhores resultados do GSA:

Número de Épocas (**nepochs**): 2

Número de Neurônios na Camada 2 (**nneurons1**): 25

Número de Neurônios na Camada 3 (**nneurons2**): 32

Accuracy: 84.44%

Especificidade: 84.44%

Sensibilidade: 89.08%

F-measure: 86.70%

AUC (OvR): 92.18%

AUC (OvO): 89.19%

Matriz de Confusão: **[33, 0, 0;**

0, 37, 0;

14, 0, 6]

Melhores resultados do PSO:

Número de Épocas (**nepochs**): 3

Número de Neurônios na Camada 2 (**nneurons1**): 23

Número de Neurônios na Camada 3 (**nneurons2**): 17

Accuracy: 42.22%

Especificidade: 42.22%

Sensibilidade: 77.57%

F-measure: 54.68%

AUC (OvR): 73.47%

AUC (OvO): 71.90%

Matriz de Confusão: **[33, 0, 0;**

37, 0, 0;

15, 0, 5]

Comparação:

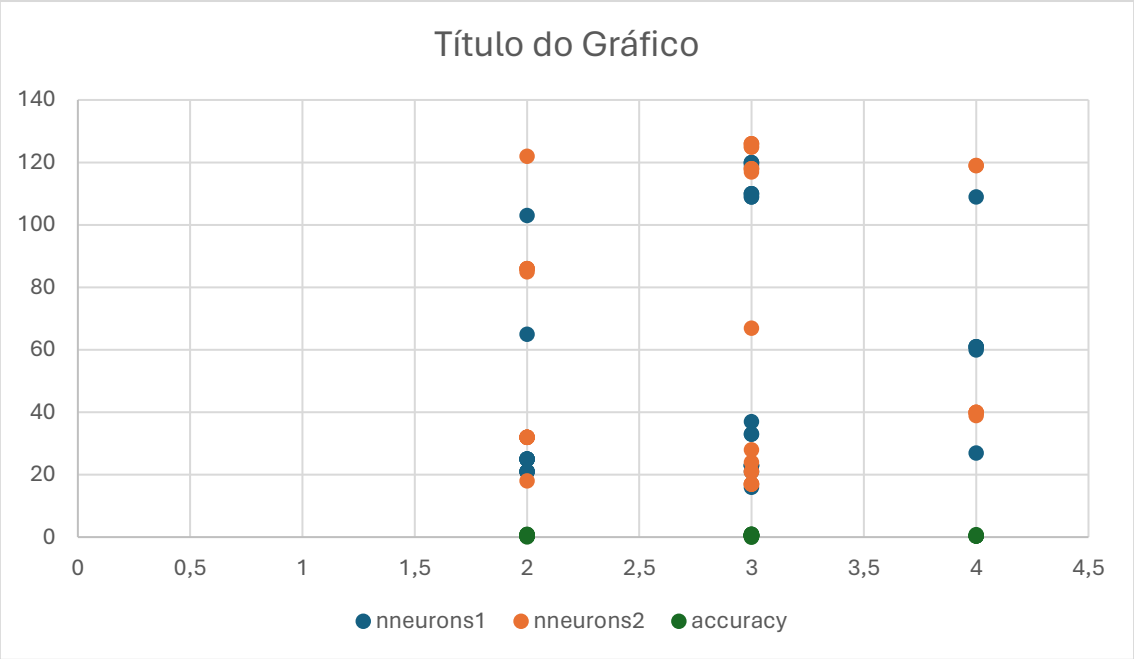
O GSA obteve uma accuracy significativamente superior em comparação com o PSO (84.44% vs. 42.22%).

Na especificidade e sensibilidade, o GSA também superou o PSO.

O PSO apresentou um desempenho inferior, especialmente na classe 1, onde a precisão foi prejudicada.

Ambos os algoritmos apresentaram uma AUC (Área sob a Curva ROC) considerável, indicando uma boa capacidade de discriminação.

Hiperparâmetros:



Com Vs Sem Transfer Learning:

Utilizando o mesmo numero de imagens no dataset.

Resultados da meta II (sem transfer learning):

GSA:

Accuracy: 54.37%

Especificidade: 54.37%

Sensibilidade: 64.29%

F-measure: 58.92%

AUC (OvR): 65.72%

AUC (OvO): 65.32%

PSO:

Accuracy: 33.57%

Especificidade: 33.57%

Sensibilidade: 82.81%

F-measure: 47.78%

AUC (OvR): 60.89%

AUC (OvO): 60.45%

Comparação:

Accuracy:

Transfer Learning superou o treino sem transfer learning.

A inclusão de conhecimento prévio da Xception contribuiu para um melhor desempenho geral.

Especificidade e **Sensibilidade**:

O GSA com Transfer Learning superou nas duas métricas em comparação com o treino sem transfer learning.

Transfer Learning apresentou vantagens, especialmente na sensibilidade.

F-measure:

Em ambas as configurações, GSA com Transfer Learning obteve um resultado mais equilibrado entre precisão e recall.

AUC (OvR e OvO):

O Transfer Learning demonstrou uma melhoria na capacidade de discriminação, indicada pelas métricas AUC.

Conjunto de dados independente:

Resultados do modelo carregado:

Accuracy: 84.44%

Especificidade: 84.44%

Sensibilidade: 88.98%

F-measure: 86.65%

AUC (OvR): 89.33%

AUC (OvO): 87.08%

Matriz de Confusão: **[[34 0 0]**

[0 34 0]

[14 0 8]]

Análise:

O modelo treinado com Transfer Learning manteve um desempenho consistente ao ser avaliado em por um conjunto de dados independente.

A accuracy de 84.44% indica uma capacidade robusta de generalização do modelo para novos dados.

As métricas de precisão, recall e f1-score mostram que o modelo é eficaz em todas as classes, com destaque para a classe 1 com resultados perfeitos.

Conclusões

Com base nos resultados obtidos, o algoritmo GSA demonstrou ser mais eficaz na procura pelos melhores hiperparâmetros para a tarefa em questão.

A utilização de Transfer Learning, neste caso, resultou em modelos com melhor desempenho em comparação com o treino sem transfer learning.

GSA com Transfer Learning destacou-se como a melhor combinação de algoritmo de otimização e abordagem de aprendizagem.

Transfer Learning é uma estratégia eficaz para tarefas de classificação de imagens, permitindo que o modelo aproveite o conhecimento retirado de conjuntos de dados maiores.

Referências

https://keras.io/api/applications/

https://keras.io/api/applications/xception/