Nome: **Leonardo Edilson Auler**	Curso: **Ciência da Computação**	Nota:
Disciplina: Fundamentos de Inteligência Artificial + Inteligência Artificial		
Docente: Filipo Novo Mór	Grau: 2	Data: 27/06/2025

Funcionamento detalhado do código:

- 1 Treinamento do Modelo (treinamento.py)

O script realiza o treinamento de uma rede neural convolucional para classificação binária: identificar se o Wally está presente ou não em uma imagem.

Fluxo do Código:

- Define os diretórios de treinamento e validação das imagens que contem o Wally e que não contem: **waldo_dataset/train** e **waldo_dataset/val**

- Carrega o modelo base MobileNetV2 sem a parte final (top), com pesos do ImageNet

- Devido treinamento de 400 épocas o mesmo se tornou extremamente demorado devido a limitações de Hardware

Salva o modelo como waldo_classifier3.h5 (Último modelo utilizado para testes funcional).

- 1.2 Inferência (inferencia_Wally.py)

- Realiza a detecção do Wally em imagens grandes usando janela deslizante (sliding window), as imagens de teste estão na pasta “Original-Images”.

Fluxo do Código:

- Carrega o modelo treinado waldo_classifier3.h5 (Melhor modelo com maior tempo de treinamento (400 épocas)).

- Carrega a imagem

- Desliza uma janela de 224x224 px com step_size=12 pixels

- Cada janela é normalizada e passada para o modelo

- Se a previsão ultrapassar o threshold (0.9), marca a detecção e soma no heatmap

Gera visualização final com:

- Detecções com retângulo verde e confiança
- Heatmap sobreposto
- Salva resultado como resultado_wally_com_heatmap.png

Instruções para executar o treinamento:

Criar ambiente virtual e instalar dependências:

1. python -m venv .venv
2. .venv\Scripts\activate
3. pip install tensorflow==2.15.0 opencv-python matplotlib seaborn

Necessário ter a seguinte estrutura de pastas contidas no repositório para treino:

```
>waldo_dataset/
        > train/
        >   >notwaldo
        >   >waldo/

        >val/
        >   >notwaldo/
        >   >waldo/
```

Para executar o código: python treinamento.py

Instruções para realizar o reconhecimento:

1. Verifique se o arquivo **waldo_classifier3.h5** existe.
2. Ajuste o caminho da imagem no script de inferência:
    a. **img_path = "original-images/ID_ESCOLHIDO.jpg"**

Para executar o código: python inferencia_Wally.py

Saídas:

- Visualização com matplotlib
- Resultado salvo como resultado_wally_com_heatmap.png

Instruções para que o professor execute o reconhecimento utilizando os arquivos de exemplo disponibilizados pelo grupo:

1. **Clonar ou copiar os arquivos enviados:**
    a. treinamento.py
    b. inferencia_Wally.py
    c. waldo_classifier3.h5
    d. Diretórios:
        i. original-images/
        ii. waldo_dataset/

2. **Instalar dependências:**

    a. pip install tensorflow==2.15.0 opencv-python matplotlib seaborn

3. **Executar o script de inferência:**

    a. python inferencia_Wally.py

Observações Finais:

Devido a limitações de hardware testes de funcionamento correto desde a aprendizagem até o teste se tornavam muito demorados, esta aprendizagem “waldo_classifier3.h5” demorou mais de dois dias para finalizar as 400 épocas, o que causou um certo atraso, fora que foi necessário criar um scrip para organizar as imagens de treino, pois o dataset não veio pronto e a maior dificuldade de todas foi fazer o ambiente reconhecer a GPU, pois sem ela seria inviável realizar o treinamento a tempo.

Foi uma grande aprendizagem trabalhar neste projeto o que também me fez ter vontade de me aprofundar mais e futuramente criar uma IA que auxilie na leitura de logs do asterisk para detecção de falhas e ajuste de problemas que possam surgir no console.