ML / IA Challange

1. Configuração do ambiente

a. Fazer o download e instalar o Miniconda3 para auxiliar na gerenciamento de ambientes próprios. Adicione o diretório ao PATH do sistema pelo Painel de Controle.

b. No terminal do Visual Studio Code, é necessário os seguintes comandos:

- conda init
- conda create -n detectron_env python=3.9
- conda activate detectron_env
- conda install pytorch torchvision torchaudio pytorch-cuda=11.8 -c pytorch -c nvidia
- pip install 'git+https://github.com/facebookresearch/detectron2.git'
- pip install opencv-python-headless pyyaml tqdm
- pip install gradio

2. Preparação do Dataset COCO 2017

a. Download do dataset através dos seguintes links:

- Conjunto de treinamento: **http://images.cocodataset.org/zips/train2017.zip**
- Conjunto de validação: **http://images.cocodataset.org/zips/val2017.zip**
- Anotações com os labels das imagens: **http://images.cocodataset.org/annotations/annotations_trainval2017.zip**

b. Conversão das imagens para formato binário:

- Através do arquivo Desafio_ML_transform_dataset.py, todas as imagens são

convertidas para o formato ".npy";

- Estrutura proposta:

```
/path/
├── train2017_converted/
├── val2017_converted/
└── annotations/
        ├── instances_train2017.json
        └── instances_val2017.json
```

3. Customização do DatasetMapper:

a. Criação do DatasetMapper Customizado: Detectron2 normalmente espera imagens em formatos como .jpg ou .png. Ao customizar o DatasetMapper, permitimos que o pipeline de dados do Detectron2 processe diretamente imagens no formato ".npy".

- A customização do DatasetMapper foi feita através do arquivo Desafio_ML_dataset_mapper.py;

4. Registro do Dataset no Detectron2:

a. Registrar o dataset no Detectron2 é necessário para que o framework possa acessar e utilizar os dados durante o treinamento e avaliação. Foi ajustado o registro para usar o DatasetMapper customizado, que processa as imagens .npy.

- O registro foi feito através do arquivo Desafio_ML_dataset_register.py;
- Para facilitar na execução dos scripts, foi adicionado esse trecho de código no arquivo Desafio_ML_model_train.py;

5. Configuração do Modelo:

a. Carregamento do Modelo: Foi utilizado o modelo pré-treinado Mask R-CNN. Isso é

vantajoso porque o modelo já foi treinado em uma grande quantidade de dados, o que pode acelerar significativamente o processo de convergência. Configurar o número de classes é importante para que o modelo se ajuste ao conjunto de dados específico (pessoas e gatos).

- A configuração do modelo e características para o treinamento foi feito através do arquivo Desafio_ML_model_config.py;
- Para facilitar na execução dos scripts, foi adicionado esse trecho de código no arquivo Desafio_ML_model_train.py;

6. Treinamento do Modelo:

a. O modelo foi treinado com o DatasetMapper customizado para que seja possível carregar imagens ".npy".

- O treinamento foi feito através do arquivo Desafio_ML_model_train.py;
- O modelo final foi salvo em output/model_final.pth;
- As métricas ao longo do treinamento podem ser analisadas em output/metrics.json;

7. Criação da Interface:

a. Para criação da User Interface foi utilizado Gradio, uma ferramenta utilizada para criar interfaces web interativas diretamente a partir do código Python. Ele simplifica o processo de criar uma UI para modelos de aprendizado de máquina.

- A interface foi feita através do arquivo Desafio_ML_UI.py;