Desafio - IA

Este desafio está dividido em 4 etapas para segmentar uma imagem para detecção de vegetação:

1. processos para lidar com imagens grandes e preparação dos dados para uso em treinamentos e inferências de modelos de IA.
2. Preparação de um dataset para treino de um modelo de IA para segmentação de imagens.
3. Treinamento de um modelo.
4. Execução da inferência utilizando modelo treinado.

Ao final do projeto do desafio, você terá como resultado um modelo de rede neural artificial treinado para segmentar uma imagem para detecção de vegetação. O modelo irá basicamente replicar o método de binarização da imagem.

A entrega deverá ser feita via repositório git de sua preferência.

Critérios de avaliação:

- Clareza e organização do código implementado
- Organização de *commits* no repositório Git
- Senso crítico para resolução de problemas de visão
    - Atenção à capacidade de generalização do modelo de IA

Etapas de Implementação

1. Quebra de imagem em blocos

A primeira etapa consiste em quebrar uma imagem TIFF em partes menores. Na prática, os ortomosaicos podem ser muito grandes, o que inviabilizaria a execução de uma inferência de rede neural. Para isso, precisa-se quebrá-los em blocos (ou janelas) menores.
Para isso, implemente um script Python que divida o arquivo de imagem original e blocos de imagens menores, salvando-os em arquivos PNG ou JPG. Ao final, espera-se que sejamos capazes de executar o script da seguinte forma:

```
python divide_orthomosaic.py --input </path/to/orto.tif> --output
</path/to/output/dir/>
```

Onde:

- `</path/to/orto.tif>` corresponde ao caminho do arquivo ortomosaico.
- `</path/to/output/dir/>` corresponde ao caminho do diretório onde as imagens divididas serão salvas.

2. Geração de Dataset

Na segunda, você deverá gerar o dataset de segmentação de imagens, o qual irá alimentar o modelo de rede neural na etapa de treinamento e validação.
Para isso, implemente um script Python que realize a binarização dos blocos das imagens, de forma que atribua-se o valor 1 a todos os pixels correspondentes a algum tipo de vegetação e 0 (zero) para os pixels que não sejam vegetação. Salve o resultado em imagem de escala cinza (PNG ou JPG) em um diretório separado.

Ao final, espera-se que sejamos capazes de executar o script da seguinte forma:

```
python binarize_images.py --input </path/to/images/dir> --output
</path/to/segmented/dir/>
```

Onde:

- `</path/to/images/dir>` corresponde ao caminho do diretório que contém as imagens RGB em blocos.
- `</path/to/segmented/dir/>` corresponde ao caminho do diretório onde serão salvas as imagens segmentadas em escala cinza, onde 0 (zero) corresponde aos pixels sem vegetação e 1 aos pixels com vegetação.

Pode escolher ou desenvolver seu próprio método de binarização da imagem. Para a binarização de imagens pode ser utilizado a própria coloração verde ou índices vegetativos, como ExG (Excess Green Index), GLI (Green Leaf Index), etc.

3. Implementação e Treinamento de Rede Neural

A terceira parte do desafio consiste em implementar uma arquitetura de Rede Neural Artificial para segmentação de imagens (pode adotar arquiteturas conhecidas também) e treiná-la. Considere quebrar esta implementação em partes para melhor organização do código.

Ao final, espera-se que sejamos capazes de executar o treinamento de um modelo de rede neural através do seguinte comando:

```
python train_model.py --rgb </path/to/images/dir> --groundtruth
</path/to/segmented/dir/> --modelpath </path/to/model.h5>
```

Onde:

- `</path/to/images/dir>` corresponde ao caminho do diretório que contém as imagens RGB em blocos.
- `</path/to/segmented/dir/>` corresponde ao caminho do diretório onde serão salvas as imagens segmentadas em escala cinza, onde 0 (zero) corresponde aos pixels sem vegetação e 1 aos pixels com vegetação.
- `</path/to/model.h5>` corresponde ao caminho modelo final salvo.

4. Inferência do Modelo

A última etapa corresponde à fase da inferência, onde o modelo treinado será usado para segmentar ortomosaicos RGB para identificação de regiões onde há vegetação ou não. Este último algoritmo será usado para testes em imagens que não foram fornecidas para o treinamento.
Considere que o ortomosaico pode ter tamanho e resolução variados.

Ao final desta etapa, espera-se que sejamos capazes de executar a inferência do modelo atraveś do seguinte comando:

```
python model_inference.py --rgb </path/to/orto.tif> --modelpath
</path/to/model.h5> --output </path/to/segmented/mask.tif>
```

Onde:

- `</path/to/orto.tif>` corresponde ao caminho do arquivo de imagem RGB a ser segmentado.
- `</path/to/model.h5>` corresponde ao caminho modelo treinado.
- `</path/to/segmented/mask.tif>` corresponde ao caminho onde será salvo o arquivo de imagem segmentado pelo modelo de IA.

5. [Extra] Vetorização do resultado

Esta é uma etapa extra e opcional. Nesta etapa, deve-se realizar a vetorização (ou *poligonização*) da máscara segmentada do ortomosaico. O resultado deve ser salvo em um arquivo GeoJSON no sistema de coordenadas EPSG:4326.
Considere incluir alguns dados estatísticos, como por exemplo a área de cobertura vegetal de cada região de vegetação.

Ao final desta etapa, espera-se que sejamos capazes de executar a vetorização atraveś do seguinte comando:

```
python vectorize_mask.py --mask </path/to/mask.tif> --output
</path/to/polygons.geojson>
```

Onde:

- `</path/to/mask.tif>` corresponde ao caminho do arquivo de imagem da máscara da segmentação.
- `</path/to/segmented/polygons.geojson>` corresponde ao caminho onde será salvo o arquivo GeoJSON dos polígonos das vegetações segmentadas.