José Matheus Carvalho Boaro
Matrícula: 2120592

Projeto Final de Programação: Ferramenta a visualização e utilização de modelos de *deep learning* em imagens

Rio de Janeiro, Brasil

Dezembro - 2022

José Matheus Carvalho Boaro
Matrícula: 2120592

Projeto Final de Programação: Ferramenta a visualização e utilização de modelos de *deep learning* em imagens

Trabalho apresentado ao coordenador do Programa de Pós-Graduação em Informática da PUC-Rio como requisito para obtenção de nota na disciplina INF2102 - Projeto Final de Programação.

Departamento de Informática

Pontifícia Universidade Católica do Rio de Janeiro

Orientador: Prof. Dr. Sérgio Colcher

Rio de Janeiro, Brasil

Dezembro - 2022

Sumário

1 Especificação do Projeto

1.1 Finalidade

Esse documento apresenta as especificações de um sistema de visualização e utilização de modelos de *deepleaning* para facilitar e auxiliar pesquisadores no teste de diversos modelos de predição em imagens, em espécifico no contexto do desenvolvimento do módulo de percepção de robôs móveis.

Robôs móveis são caracterizados por 4 principais áreas, locomoção, percepção, cognição e navegação. Onde a locomoção diz respeito a parte físicas do robô, rodas, pernas mecânicas e etc. Percepção diz respeito aos sensores que dão ao robô informações para serem utilizadas.Cognição, é o módulo responsável por processar as informações obtidos por sensores e devolver informação útil para a tomada de decisão do robô. Por fim, temos a navegação, que seria a área responsável pela lógica de locomoção do robô.

A percepção de um robô é caracterizada pela informações adquiridas do ambiente por meio de sensores que subsequentemente são processadas e transformadas em informações úteis. A principais áreas envolvidas são *machine learning* e visão computacional na tentativa de mapear, representar, posicionar e localizar o robô dentro de um determinado ambiente, controlado ou não.

Robôs moveis podem ter seu sensores classificados entre internos e externos, onde os internos mensuram valores interno ao robô como, voltagem da bateria, velocidade, entre outros e os externos mensuram coisas relacionadas ao ambiente em que o sensor está inserido, como microfones, câmeras e etc. . .

No contexto do projeto "Ad Doc Teamworks", que investiga a interação entre humanos e robôs em tarefas conjuntas, essa ferramenta será útil para testar possíveis modelos que poderão ser utilizados no sistema de percepção do robô móvel. Nesse contexto, o usuário poderá escolher modelos de detecção de objetos, descrição de imagens e modelos de *question and answer*, responsáveis por responder perguntas referentes a imagem utilizada. Logo, como resultado, espera-se obter os objetos detectados na imagem, uma descrição automática da imagem e a resposta para uma pergunta realizada pelo usuário. Além disso será possível exportar todas essas informações para possível utilização do usuário, podendo ser uma forma de alimentar o módulo de percepção de um robo móvel com informações sobre o cenário visto.

1.2 Escopo

O escopo deste projeto é a produção de um sistema de visualização e utilização de modelos de *deepleaning*. O programa deve auxiliar o usuário a utilizar modelos pré-definidos além de mostrar ao usuário o resultado obtido a partir da inferência de cada modelo utilizado. Os resultados incluem objetos detectados, descrição automática da imagem e resposta para uma pergunta realizada pelo usuário. Por fim o usuário pode exportar os resultados em formato ".json".

1.3 Requisitos

1.3.1 Requisitos Funcionais

O requisitos funcionais do sistemas foram elicitados baseado na finalidade geral do programa, permitir e facilitar a utilização e visualização de modelos de *deep learning*.

- **RF01** - A ferramenta deve permitir que o usuário carregue uma imagem de qualquer diretório do seu computador;
- **RF02** - A ferramenta deve permitir que o usuário selecione dentre os modelos cadastrados, um modelo de detecção de objetos que será utilizados para inferência.
- **RF03** - A ferramenta deve permitir que o usuário selecione dentre os modelos cadastrados, um modelo de descrição de imagens que será utilizados para inferência.
- **RF04** - A ferramenta deve permitir que o usuário selecione dentre os modelos cadastrados, um modelo de resposta que será utilizados para inferência.
- **RF05** - A ferramenta deve permitir que o usuário possa digitar uma pergunta a ser respondida pelo modelo de resposta durante a inferência.
- **RF06** - A ferramenta deve mostrar todos os objetos detectados individualmente durante a etapa de inferência em forma de lista horizontal.
- **RF07** - A ferramenta deve mostrar todos os objetos detectados durante a etapa de inferência marcados na imagem original passada.
- **RF08** - A ferramenta deve mostrar todos os objetos detectados durante a etapa de inferência marcados na imagem original fornecida pelo usuário.
- **RF09** - A ferramenta deve exibir a descrição automática gerada pelo modelo durante a inferência.
- **ARF10** - A ferramenta deve exibir a resposta gerada pelo modelo durante a inferência.

- **ARF11** - A ferramenta deve permitir o usuário exportar todos os metadados dos modelos de inferência utilizados, bem como a predição de cada modelo.

1.3.2 Requisitos Não-Funcionais

Dentre os requisitos não-funcionais que a ferramenta deve apresentar, listam-se:

- **RNF01 - Performance:** Refere-se ao tempo de resposta durante o uso das funcionalidades fornecidas pelo sistema;
- **RNF02 - Usabilidade:** Refere-se à facilidade de aprendizagem e uso do sistema;
- **RNF03 - Confiabilidade:** Deve-se buscar baixa taxa de erro durante o uso do sistema e robustez para resolvê-los;
- **RNF04 - Padrões:** Refere-se à conformidade do desenvolvimento do sistema com padrões de software em geral;
- **RNF05 - Compatibilidade:** Refere-se à compatibilidade de uso do sistema em OS Windows;

2 Arquitetura e Projeto

O projeto foi desenvolvido utilizando a biblioteca PyQT5, que permite a criação de aplicações desktop. Para toda a parte de *machine learning* fora utilizada bibliotecas como HuggingFace, que facilita a utilização e distribuição de modelos de *deeplearning*, onde em mais baixo nível se utilizou a biblioteca PyTorch para criação e de desenvolvimento dos modelos de deeplearning utilizados. [1] O projeto foi totalmente desenvolvido na linguagem Python versão 3.8.

A Figura 2 mostra a árvore de arquivos do programa. O projeto possui um diretório principal contendo os subdiretórios de interface, modelos e dados. O diretório de interface contém o arquivo referente a parte gráfica da aplicação. O diretório de modelos possui as implementações abstratas dos modelos de deeplearning utilizados. O diretório de dados possui as implementações das classes especificas de cada modelo de deeplearning e também a classe principal chamada "Anlyzer.py", que implementa e orquestra todo o funcionamento do programa. A Figura ilustra a árvore de aquivos do programa.

```
metadata_visualizer/
    main.py
    data/
        analyzer.py
        fastrcnn.py
        utils.py
        VilT.py
        vit_gpt2.py
        yolo.py
        config/
            fastrcnn/
                faster_rcnn_r50_fpn_carafe_1x_coco.py
                faster_rcnn_r50_fpn_carafe_1x_coco_bbox_mAP-0.386_20200504_175733-385a75b7.pth
            yolo/
                yolov3_d53_mstrain-608_273e_coco.py
                yolov3_d53_mstrain-608_273e_coco_20210518_115020-a2c3acb8.pth
    interface/
        mainpage.py
        mainpage.ui
        utils.py
    metadata/
        0_person.jpg
        1_person.jpg
        2_person.jpg
        3_person.jpg
        teste.png
    model/
        deeplearning_models/
            abstract_model.py
            image_captioning_model.py
            object_detection_model.py
            question_and_answer_model.py
    test/
        run_test.py
```

Figura 1 – Estrutura das pastas no projeto.

Dentro do diretório também temos a pasta de metadados, utilizada para armazenar os dados intermediários que são criados durante a utilização do programa. Nele ficam o recorte da imagem utilizada e dele que a interface irá consumir os dados para serem

[1] https://huggingface.co/

exibidos na aplicação durante o funcionamento. Por fim temos a pasta "test", nela está especificado alguns testes de unidade para verificação da integridade das funções utilizadas na aplicação.

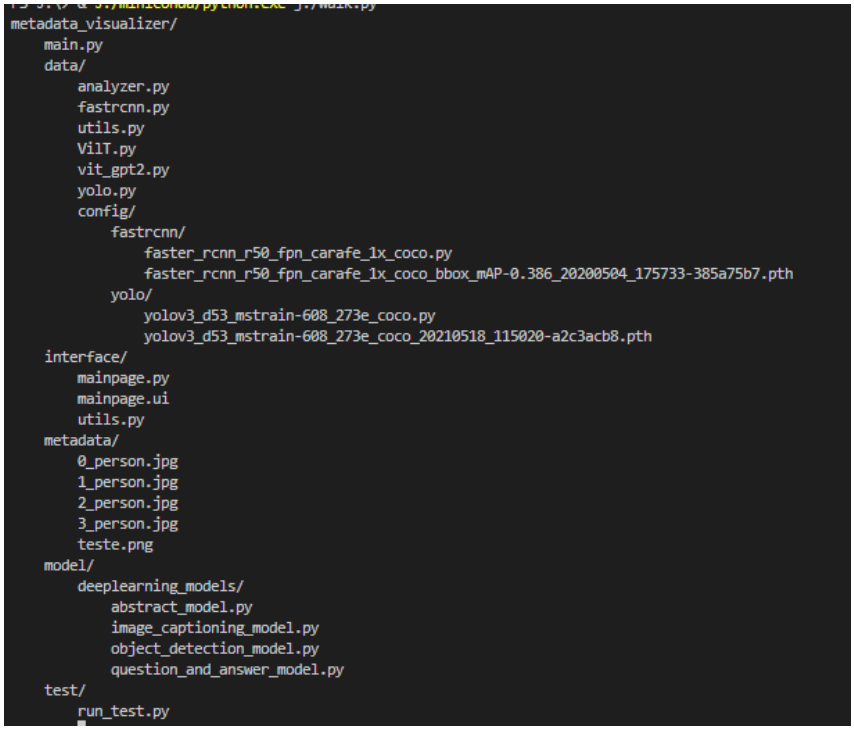


Figura 2 – Estrutura das pastas no projeto.

2.1 Descrição das etapas da aplicação

O módulo principal do sistema, chamado "Analyzer"tem o seu funcionamento descrito através da Figura 3, que mostra um diagrama de atividade do fluxo principal da aplicação.

Já a Figura 4 , ilustra as atividades do módulo de detecção de objetos. Esse módulo tem como principais ações a detecção e o recorte dos objetos detectados para posterior utilização pela aplicação.

A Figura 5, por sua vez, ilustra as atividades do módulo de detecção de descrição automática de imagens.

Por fim, a Figura 6, ilustra as etapas da execução do módulo de perguntas e respostas.

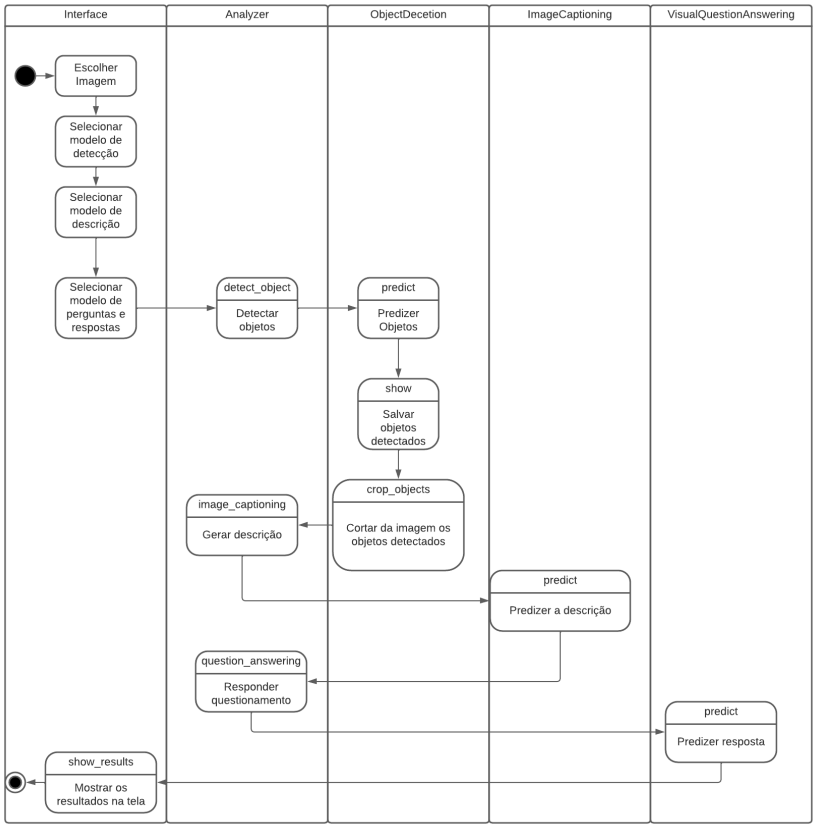


Figura 3 – Diagrama de atividades do fluxo principal da aplicação.

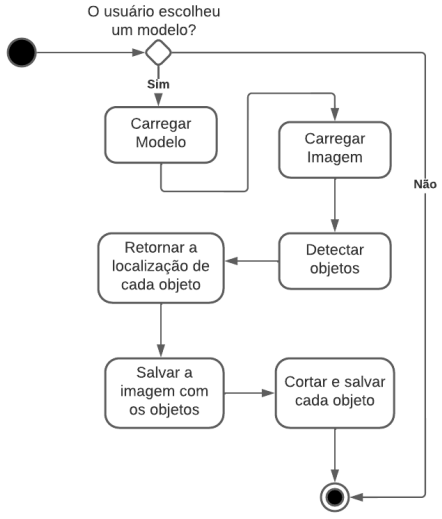


Figura 4 – Diagrama de atividades do modulo de detecção.

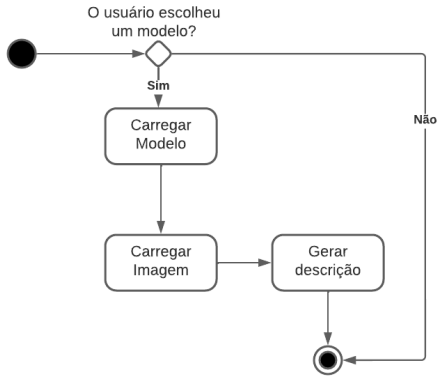


Figura 5 – Diagrama de atividades do modulo de descrição automática de imagens.

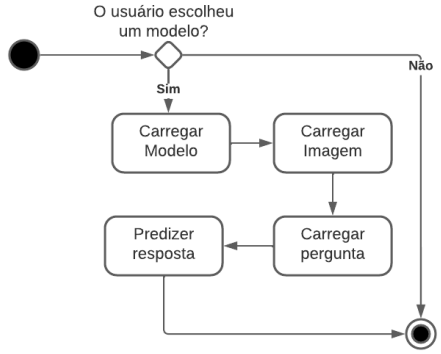


Figura 6 – Diagrama de atividades do modulo de perguntas e respostas.

2.2 Considerações Finais

O código fonte da ferramenta se encontra no Github [2]. Todas as dependências e bibliotecas que precisam ser instaladas para o bom funcionamento da aplicação se encontram no arquivo *requirements.txt*.

[2] https://github.com/

3 Testes

Esse capitulo descreve os testes feitos para atestar a boa funcionalidade das função criadas. Nesse sentido, foram testadas as funções de detecção de objetos, descrição automática de imagens, resposta de perguntas sobre a imagem, ao todo foram realizados testes, que serão descritos abaixo. Os testes foram desenvolvidos na linguagem Python utilizando a biblioteca padrão da linguagem "unittest"que implementa testes automáticos.

Abaixo segue a descrição de cada funcionalidade testada:

- **Detecção de objetos**

  Esse teste consistiu em atestar a boa funcionalidade do módulo de detecção de objetos. Fora testada a criação e predição do modelo Detr (CARION et al., 2020), além do teste quando o usuário não escolhe nenhum modelo. Para o primeiro caso, espera-se que o modelo retorne uma lista com a localização de cada objeto detectado. No segundo caso, espera-se que seja retornado um valor nulo (None).

- **Descrição automática de imagens**

  De forma similar ao teste anterior, o módulo de descrição automática também fora testado. Ao todo, parece esse módulo, também foram realizados 2 testes, o primeiro, direcionado ao modelo VitGPT2 (KUMAR, 2022), onde fora testado a instanciação e predição do modelo, esperando como retorno uma *string* com a descrição vista na imagem. O outro teste foi realizado simulando o caso de quando o usuário não escolhe nenhum modelo de descrição automática de imagens, onde deve ser retornado o nulo.

- **Perguntas e respostas automáticas baseadas na imagem** Ainda de forma análoga, fora testado o modelo de perguntas e respostas. Onde quando escolhido o modelo VilT (KIM; SON; KIM, 2021), espera-se retornar uma *string* com o a resposta para a pergunta realizada pelo usuário. Além desse teste, fora realizado o teste de quando o usuário não escolher um modelo e quando o usuário não fornece uma pergunta. Para os dois últimos, a resposta esperada é o valor nulo.

```python
class AnlyzerTest(unittest.TestCase):

    def testObjectDetectionYolo(self):

        self.assertEqual(type(objectDetectionTest('Yolo','../IMG_3881.JPG')),list)

    def testObjectDetectionNone(self):

        self.assertEqual(objectDetectionTest('None','../IMG_3881.JPG'),None)

    def testImageCaptioningVitGPT2(self):

        self.assertEqual(type(ImageCaptioninTest('VitGPT2','../IMG_3881.JPG')), str)

    def testImageCaptioningNone(self):

        self.assertEqual(ImageCaptioninTest('None','../IMG_3881.JPG')[0],None)

    def testQuestionAnsweringVilt(self):

        self.assertEqual(type(QuestionAnsweringTest('VilT','../IMG_3881.JPG')), str)

    def testQuestionAnsweringNone(self):

        self.assertEqual(QuestionAnsweringTest('None','../IMG_3881.JPG'),None)

    def testQuestionAnsweringQuestionFalse(self):

        self.assertEqual(QuestionAnsweringTest('None','../IMG_3881.JPG',False),None)
```

Figura 7 – Funções de testes desenvolvidas

```
.
----------------------------------------------------------------------
Ran 7 tests in 25.841s

OK
```

Figura 8 – Resultado da execução de todos os testes.

4 Documentação para o Usuário

A aplicação desenvolvida tem como principal objetivo o teste de algoritmos de deeplearning para as tarefas de detecção de objetos, descrição de imagens e perguntas e respostas, para facilitar na inspeção visual do desempenho dos possíveis modelos escolhidos. Como principal usuário fora pensado nos pesquisadores do projeto "Ad hoc teamworks"que buscam investigar a interação de humanos e robôs moveis em tarefas conjuntas sem pre-definições de regras ou protocolos. Logo, é necessário que o robô consiga compreender o ambiente a sua volta, seja baseado na descrição da cena, ou na detecção de objetos e pessoas. Espera que o usuário tenha conhecimentos básicos de funcionamento do pacote git, assim como do framework de gerenciamento de ambientes Anaconda.

Primeiramente, o usuário deverá realizar um clone do projeto. Para isso, se faz necessário ter o git instalado na máquina do usuário. O git pode ser instalado através de diversas distribuições, como exemplo deixarei o seguinte link.

O usuário então deve instalar todas as bibliotecas e dependências presentes no arquivo "requiriments.txt". Pode ser realizado em um ambiente virtual ou em um ambiente anaconda. O usuário deverá rodar o seguinte comando: "pip install -r requirements.txt"

Por fim, para iniciar o programa o usuário deverá executar o seguinte comando "python main.py". Espera-se que após a execução do comando, a tela ilustrada na Figura 9 apareça, possibilitando a utilização da aplicação.

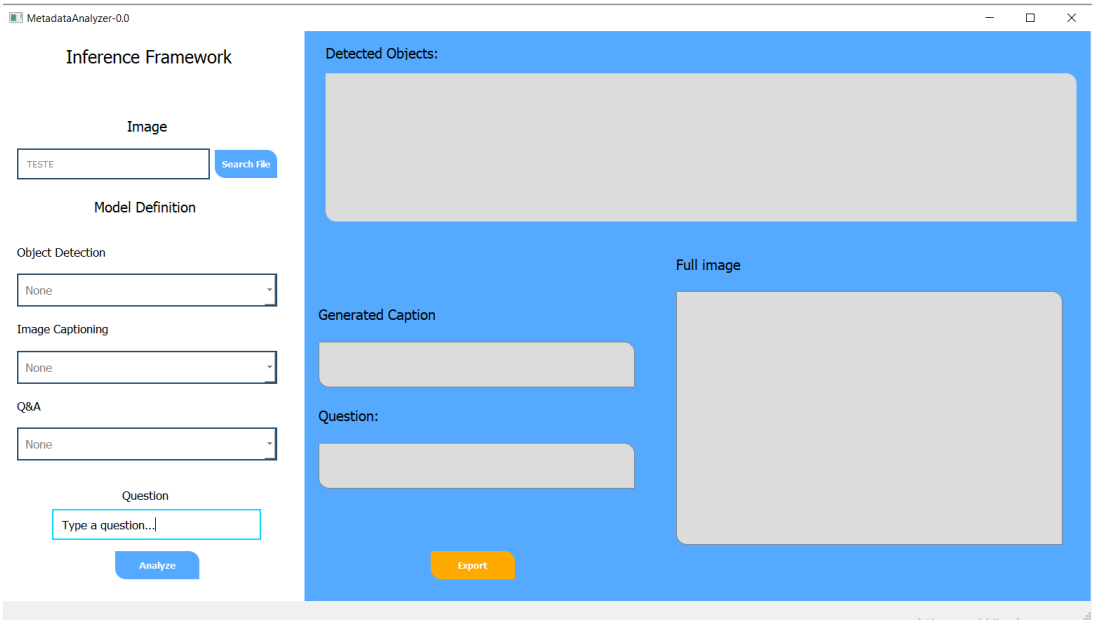


Figura 9 – Tela inicial.

Ao clicar em "search file", o usuário poderá navegar pela arvore de arquivos do

computador e escolher uma imagem para utilizar na aplicação. A Figura 10 ilustra o comportamento esperado.

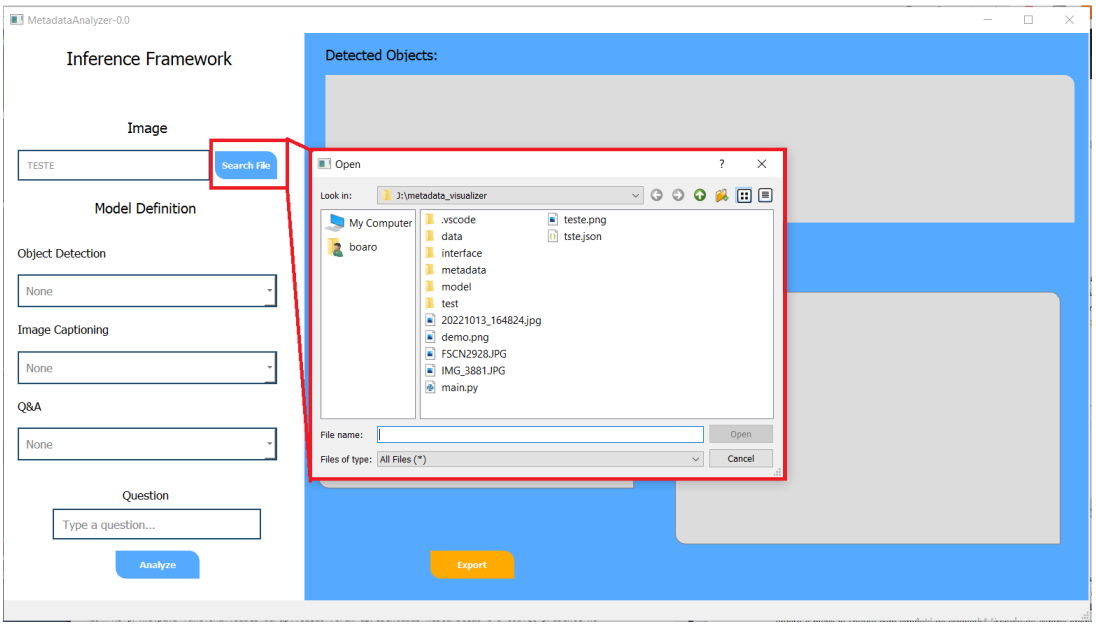
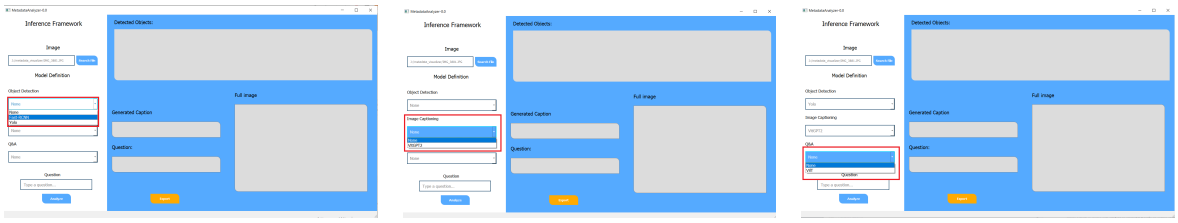


Figura 10 – Arvore de arquivo.

O usuário poderá selecionar então um dos modelos cadastrados para detecção de objetos, descrição automática e perguntas e respostas baseadas na imagem. A Figura 11 abaixo indicam os locais onde cada ação mencionada na oração anterior acontece.



(a) (b) (c)

Figura 11 – (a) Seleção do modelo de detecção de objeto. (b) Seleção do modelo de descrição automática. (c) Seleção do modelo de pergunta e resposta.

Após a seleção, o usuário deve apertas o botão "Analyze". Um exemplo de resultado após análise pode ser observado através da Figura 12. Onde em vermelho estão ressaltadas as áreas que mostram o resultado de cada modelo em suas respectivas tarefas.

Por fim, o usuário tem a opção de exportar os resultados de cada modelo, junto com suas definições em um arquivo em formato .json. Para tanto, basta clicar no botão "export", onde uma janela parecida a árvore de arquivos do sistema será mostrada, igual a ilustrada pela Figura 10, podendo escolher o local e nome do arquivo a ser exportado.

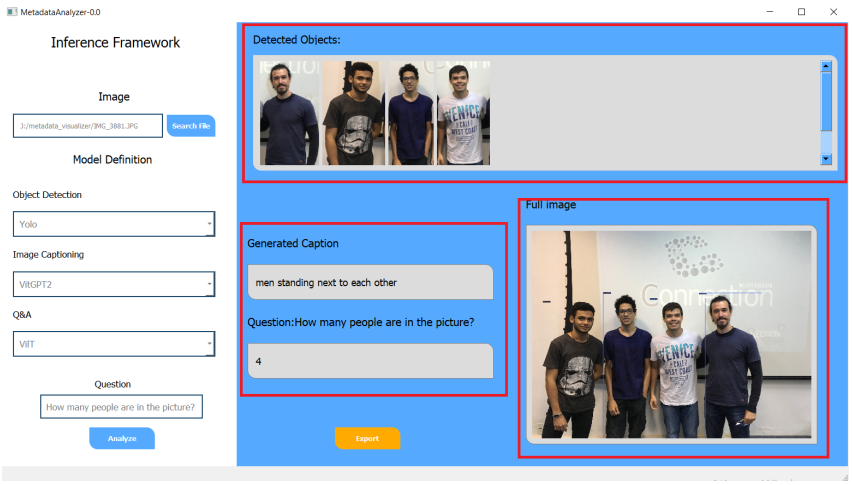


Figura 12 – Resultado após a execução da aplicação.

As principais funcionalidades da aplicação foram apresentadas nesta seção e o código presente no GitHub[1] está bem comentado. Entretanto, caso tenha alguma dúvida, entre em contado pelo *e-mail*: matheusboaro@gmail.com.

[1] https://github.com/matheusboaro/metadata_vis

Referências

CARION, N. et al. *End-to-End Object Detection with Transformers*. arXiv, 2020. Disponível em: <https://arxiv.org/abs/2005.12872>. Citado na página 10.

KIM, W.; SON, B.; KIM, I. *ViLT: Vision-and-Language Transformer Without Convolution or Region Supervision*. arXiv, 2021. Disponível em: <https://arxiv.org/abs/2102.03334>. Citado na página 10.

KUMAR, A. The illustrated image captioning using transformers. *ankur3107.github.io*, 2022. Disponível em: <https://ankur3107.github.io/blogs/the-illustrated-image-captioning-using-transformers/>. Citado na página 10.