Universidade Federal de Uberlândia
Pedro Henrique Faria Teixeira

Inteligência Artificial

Uberlândia - MG
2019

Trabalho Final: Detecção de Face e documento.

Objetivo:

- Fazer a detecção de faces e documentos em uma imagem para guardar em um banco de dados junto com as predições da imagens e seu respectivo feedback.

Definição:

- Este projeto é um classificador de imagens, identificando que há uma ou mais faces e um documento na imagem, realizando ao final da predição da imagem um feedback positivo ou negativo da imagem para armazenar no banco de dados e criar um repositório treinado de imagens válidas e inválidas para a diferenciação da **IA.** Esse processo de diferenciação e chamado de **Mechanical Turk** (um treinamento autônomo que se aproveita das imagens relatadas por outras pessoas como sendo certas ou erradas de que é aquilo que se pede, por exemplo: um carro em uma imagem). Uma imagem só é aceita neste processo de seleção se ela tiver um rosto ou mais, que pode ser só o da pessoa, ou então o da pessoa e o presente no documento. Uma imagem é rejeitada se não possui nenhuma face, ou então não possui nenhum documento.

Tecnologias:

- Linguagem de programação **Python** e suas bibliotecas para o backend e scripts de detecção de face e objeto.
- Backend criado em Flask para a integração dos serviços de detecção com o frontend.
- Junto com **Tensorflow** e **OpenCV** para que o treinamento das imagens fossem possíveis.
- Bibliotecas mais importantes: numpy, cv2, matplotlib, opencv_python, tensorflow, pickle, PIL, pandas, scikit-learn, jupyter, Flask, keras.
- Coco Trained Model **:** faster_rcnn_inception_v2_coco para o treinamento das imagens.
- LabelImg: para a circulação dos documentos em cada imagem servidas de treinamento.
- Anaconda: para a criação do ambiente e instalação das dependências.
- TensorFlow Object Detection API: usada para o treinamento das imagens, é uma API do tensorflow que possui vários scripts de detecção de imagens. Ele te possibilita criar sua própria detecção de objetos como também fornece algumas detecções mais básicas. Pode-se fazer o download em: https://github.com/tensorflow/models. O a versão usada neste projeto do tensorflow foi a **tensorflow_gpu 1.12.0.**

- Cuda e cuDNN: tensorflow não roda sem essas ferramentas.

Como foi feito:

- Após baixar todas as ferramentas necessárias, foi criado um ambiente virtual com o Anaconda e então baixada todas as depêndias. Logo após isso foi usado o LabelImg para circulação dos documentos em cada imagem para gerar o xml que contém toda as métricas do documento como : altura, largura etc. Após gerar o arquivo xml de cada imagem, aplica-se sobre eles o script xml_to_csv.py para gerar um arquivo CSV com as métricas, gerado o arquivo csv aplicar o script generate_tfrecord.py, que irá gerar arquivos .record necessários para o treinamento, em seguida cria um label.pbtxt para descrever quais os objetos serão detectados. Com tudo isso pronto temos que rodar o script de treinamento train.py localizado no repositório do tensorflow, e esperar até que chegue a um percentual de no mínimo 0.05 por cento de erro no treinamento das imagens. Após acabar o treinamento será gerado um inference_graph que é o gráfico da predição em todas as imagens com um curva rente ao eixo X que é a minimização do erro. Com isso, pegamos esse gráfico e o faster_rcnn_inception_v2_coco e utilizamos para a detecção do objeto com o arquivo object_detect_image.py.
- Para a detecção de faces, foi utilizado o OpenCv e é a parte mais simples do projeto, simplesmente a gente precisa apenas baixar o diretório cascades para utilizá-lo na detecção de faces, é uma API já pronta que dado um repositório de imagens ele faz a detecção das faces e gera as métricas para que possa ser usado em novas imagens. O script de treinament e o face_train.py que irá gerar o trainner.yml e o labels.pickle. Gerado os dois é só aplicar no script file_detect_face.py e pronto, sua detecção de faces está pronta.