07.04.2024

People detection and removal

Team #1 - Adi Finkelstein & Yuval Rubinstein

Introduction

The ultimate objective of our project is to identify and replace individuals within an image with the background seamlessly. We have divided the task into two distinct objectives, each assigned to different teams:

Objective 1: Detect People and Return their Masks (Team 1)

The first objective is to train deep learning models to accurately detect individuals in images and generate corresponding masks delineating their outlines, crucial for the mission's success in seamlessly replacing people with background elements.

Objective 2: Replace Masks with Background (Team 2)

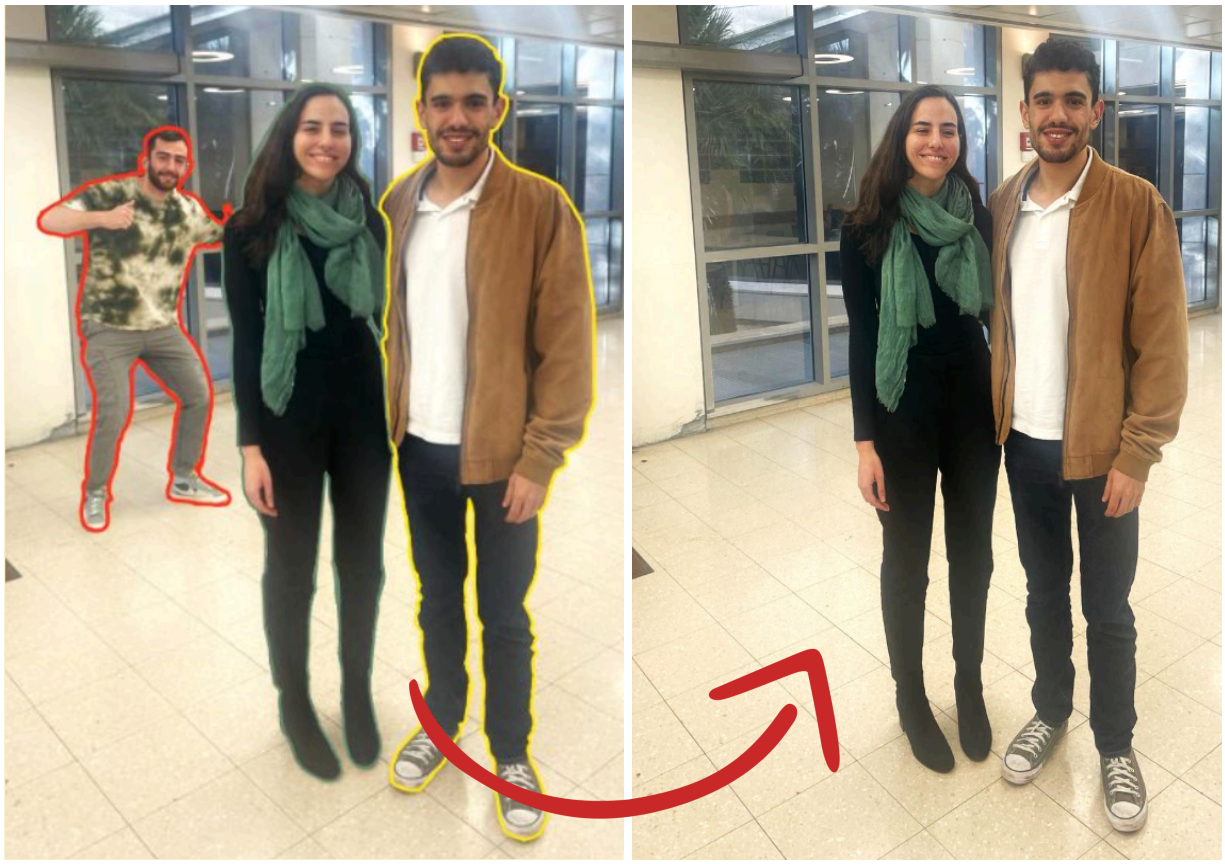
The second objective involves taking the generated masks and corresponding images, and seamlessly replacing the masked regions with background elements.

In this report, we will focus on discussing the progress and methodologies employed to fulfill the first part of the final mission, namely, the detection of people within images and the generation of corresponding masks.

Project Overview

Initial objective

At the inception of our project, our primary goal was to develop a system capable of seamlessly replacing a single person within a photograph. Additionally, we aimed to offer rotation suggestions for alternative individuals to be placed within the same photo.



Challenges Encountered

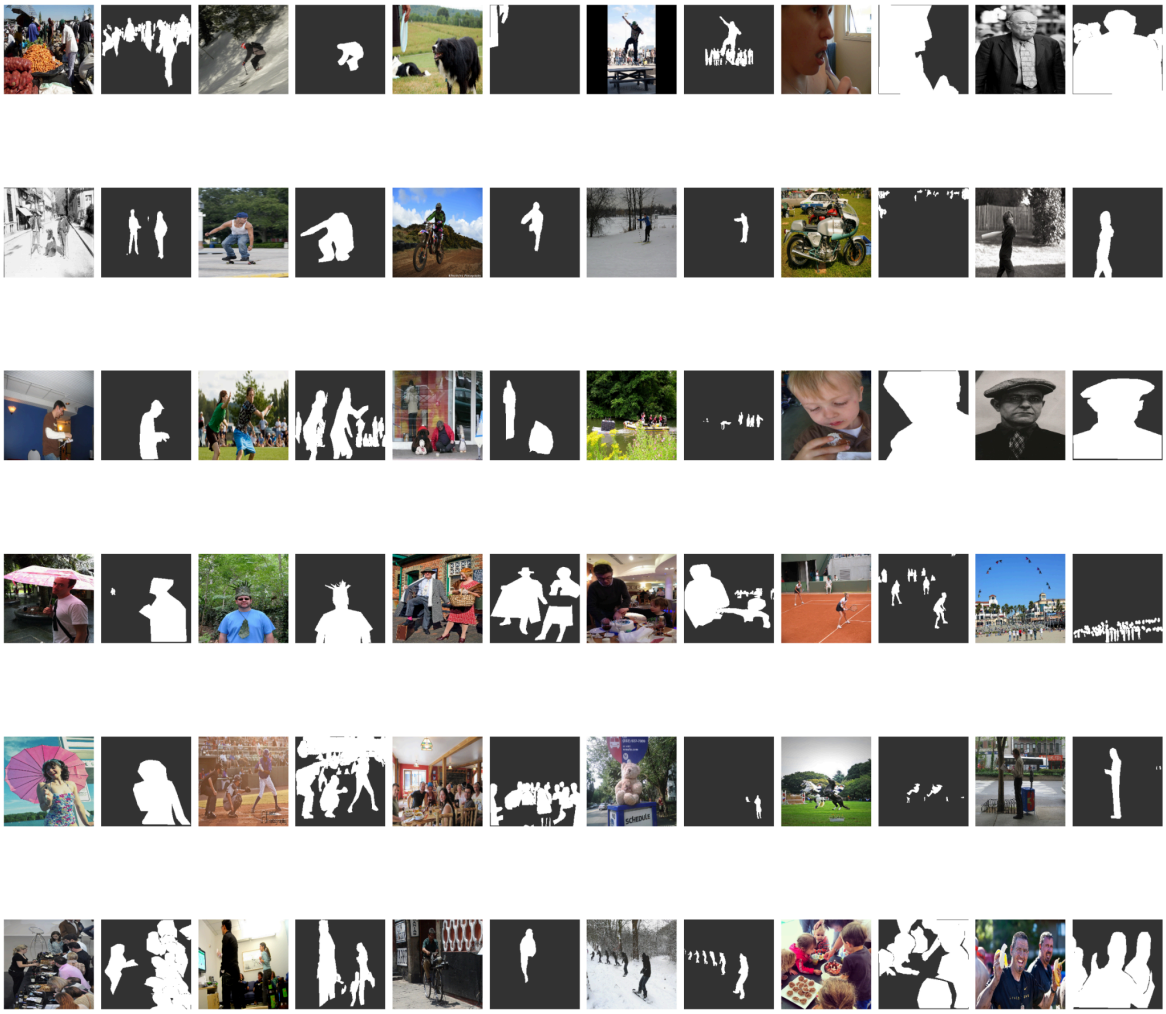
However, during the initial phases of development, we encountered a significant hurdle. Despite our efforts, we were unable to locate a suitable dataset containing the specific type of data required to train our model for this task. This absence of relevant data compelled us to reconsider our approach.

Adaptation and Refocusing

As a result of the unavailability of appropriate datasets, we made the strategic decision to pivot our project's mission. Instead of solely focusing on replacing a single individual within a photo, we broadened our scope to encompass the capability of replacing all individuals present within the frame of the photograph.

Dataset Acquisition and Preprocessing

After conducting extensive research, we successfully identified a dataset comprising over 60K images along with corresponding masks, already meticulously categorized. Leveraging the COCO dataset, we specifically filtered images falling under the 'people' category and standardized their sizes through resizing. This meticulous preprocessing ensured consistency across all images, laying a solid foundation for subsequent model training.



Code Organization and Reusability

Recognizing the collaborative nature of our project, which involved two distinct teams, we prioritized code readability, modularity, and reusability. We encapsulated various functionalities within reusable functions, accompanied by detailed documentation.
This approach not only facilitated seamless collaboration between teams but also enabled efficient troubleshooting and iteration throughout the development process.

Performance Optimization

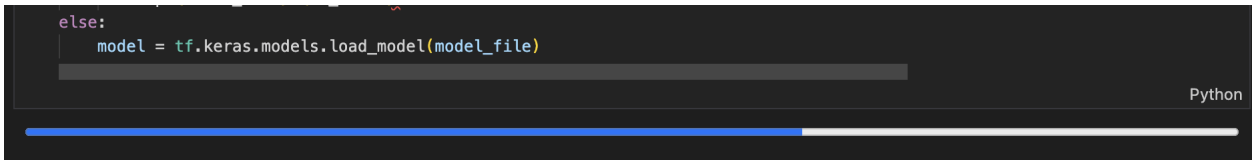
Memory Management

During model training, we encountered significant challenges related to high RAM usage, particularly when loading thousands of images into memory. To mitigate this issue and prevent program crashes, we implemented an efficient data generator. This iterator dynamically loads images in batches, thereby minimizing RAM usage and ensuring smooth execution, even within resource-constrained environments such as the free tier of Google Colab.

```python
def data_generator(
    batch_size: int,  # Amount of pictures that will be stored to the RAM each iteration
    data_type: DataType,  # train / test / validation, using pre-stored image-ids
    max_progress: int = None,  # The cap of the progress-bar
    epochs: int = 1,  # Amount of epochs to train the model
    hide_progress_bar: bool = False  # Hide / Show the progress-bar
):
```

Progress Visibility

To enhance visibility and track the progress of model fitting, we integrated a progress bar within the data generator. This feature provides real-time feedback, indicating the percentage of data processed during each iteration.



Optimized Data Retrieval

Initially, we encountered inefficiencies when each batch iteration called the API 'batch' times to retrieve images from the COCO server. To mitigate CPU overhead and significantly reduce processing time, we adopted a proactive approach. We pre-downloaded the entire dataset and stored it as a compressed ZIP file in our drive. This strategic decision not only slashed data retrieval time by over 90% but also streamlined the overall workflow. The subsequent unzip operation incurred minimal overhead, taking only a fraction of the time compared to repeated API calls.

Data Preprocessing and Handling

Outliers Handling

We encountered an issue where some of the pictures had masks that were too small, meaning that the person inside the picture occupied less than 1% of the image. To address this, we ignored these pictures but opted not to skip them entirely. Instead, we replaced them with images that were unused by the train, test, and validation datasets. We achieved this by saving a large portion of unused image IDs and swapping them while utilizing the cache we implemented. This approach ensured that we maintained the desired number of images for training (*MAX_NUM_IMAGES*).

Model Training and Evaluation

In our pursuit of the best model, we trained five models and searched for the best Intersection over Union (IOU). Each model performed predictions on the test data, and the results were analyzed. To optimize efficiency and ensure that our progress was not lost, we saved the weights of all trained models to a file, which was automatically transferred to our drive. This precautionary measure ensured that in the event of a session timeout or data clearance, we retained the valuable model weights.

Model Performance Comparison:

After training all models, we obtained the following results:

- ➔ Original model:
  - ◆ Accuracy: 0.9157
  - ◆ **Mean IOU: 0.5708**
  - ◆ Loss: 0.3175
- ➔ ResNet:
  - ◆ Accuracy: 0.8810
  - ◆ **Mean IOU: 0.6365**
  - ◆ Loss: 0.2837
- ➔ ResNet2:
  - ◆ Accuracy: 0.9003
  - ◆ **Mean IOU: 0.7084**
  - ◆ Loss: 0.2269
- ➔ VGG:
  - ◆ Accuracy: 0.8820
  - ◆ **Mean IOU: 0.6607**
  - ◆ Loss: 0.2601
- ➔ Unet (pix2pix):
  - ◆ Accuracy: 0.9406
  - ◆ **Mean IOU: 0.8168**
  - ◆ Loss: 0.1480

Based on the results, the **Unet (pix2pix)** model demonstrated the highest accuracy (0.9406) and mean IOU (0.8168) while showing the lowest loss (0.1480), thus being selected as the optimal model.

Integration and Handover:

Finally, to complete our integration, we saved the results of all test data prediction mask files into a zip archive and handed it over to team #2 for further analysis and utilization.