Final Report - Project #12 Cartoon Character Extraction

Overview

In this project we attempted to develop an algorithm which gets a video as an input, and tries to detect, extract, and classify cartoon characters into various poses.

We implemented this project in python using YOLO object detection code with help from open source. Initially we detected Rick and Morty (cartoon characters), using YOLO, each with a single trained model. We created our own dataset for the training, and we used python scripts to extend it.

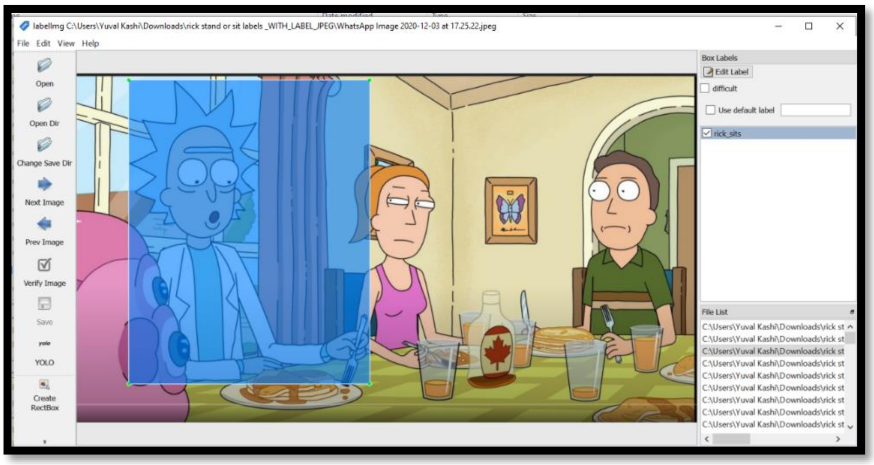
In the next phase we cropped the rectangles surrounded Rick and Morty detected characters.
finally, we classify each cropped frame into one of two poses stand or sit, using two more models, one for each character.

BGU google drive link to folder is here:
https://drive.google.com/drive/folders/1VQOYHOUQELEs8KHYhA1ekBTdM15rvUZM?usp=sharing

Training:

Generate Datasets

At first, we searched for Rick and Morty images, then we used "LabelImg" for labeling our characters on each image.



Our first training attempt failed due to the small number of images.
To solve this problem, we developed two scripts: the first script made a darker copy of each image on our dataset, the second one made a brighter copy.
After making our dataset three times bigger, we managed to get excellent results on the detection phase.

LableImg software: LabelImg (tzutalin.github.io)
Our datasets:
https://drive.google.com/drive/folders/1uyvMTTuP-sh4__OVTybHRw_nGBVougcs?usp=sharing

In order to make it simple, you can use these codes we developed.
The codes make it easy to create big database from small number of images:
(You can find them in the folder “Help functions”)

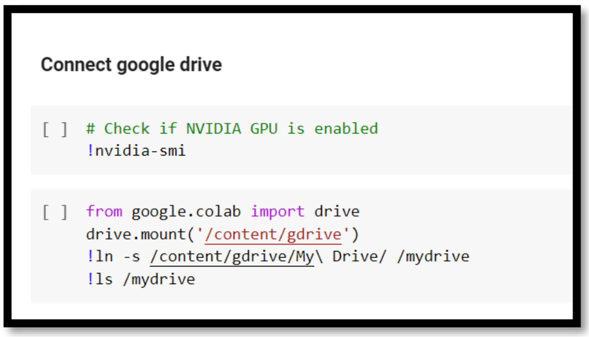
- **change files names.py**:
  Change all your images file name to start from index 1 to num-files. You can edit and enter a string before the index.
- **rename PNG to JPEG.py**:
  In order to train correctly you would like all your images to be jpeg or another all the same format name
- **make_dark_bright_images.py**:
  Enter your path to the directory with the images and a path with the directory name you want to put the new generated images.
- **add_txt_files_labels.py**:
  Enter your path to the directory with the images and a path with the directory name you want to previously put the new generated images. It will copy and generate all txt labels files.

Training Using "Google Colab" Collaboratory:

The training phase includes six steps, which can be done effectively using our .ipynb script **Train_YoloV3_1_class_.ipynb**:

Here are various instruction to run it:

1. Connect to Google Drive.

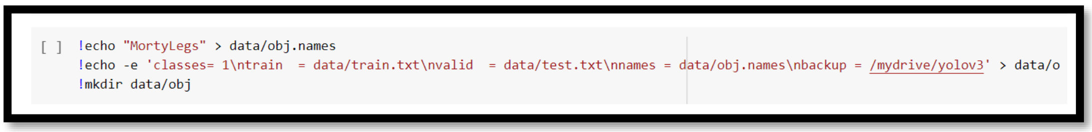


Connect google drive

```
[ ] # Check if NVIDIA GPU is enabled
    !nvidia-smi
```

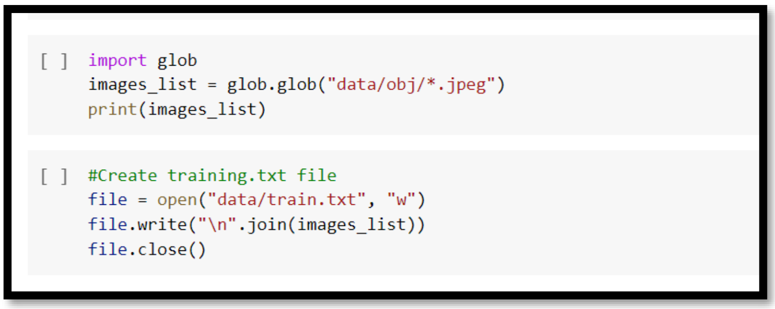
```
[ ] from google.colab import drive
    drive.mount('/content/gdrive')
    !ln -s /content/gdrive/My\ Drive/ /mydrive
    !ls /mydrive
```

   After success you can run all.
2. Clone the Darknet - the framework which we use to train YOLO.
3. Compile Darknet using Nvidia GPU.
4. Configure Darknet for training YOLO. (commands 2-3)
   Keep on mind, we train only one class, the name of it doesn't matter on the training but you can change it here in the first line:



```
[ ] !echo "MortyLegs" > data/obj.names
    !echo -e 'classes= 1\ntrain  = data/train.txt\nvalid  = data/test.txt\nnames = data/obj.names\nbackup = /mydrive/yolov3' > data/o
    !mkdir data/obj
```

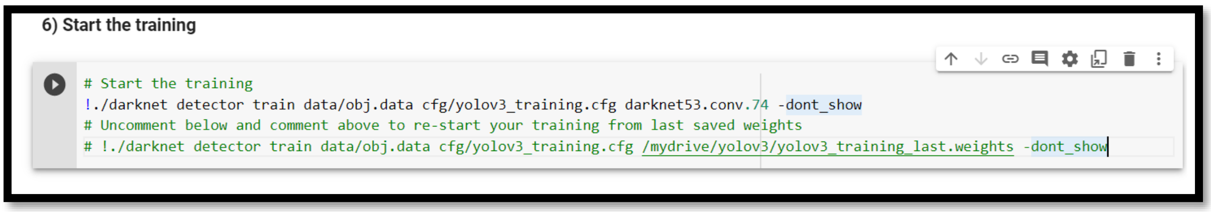
5. In step 4 the code run the DB we created.
   Be sure you put our dataset names images.zip of labeled images with txt labels which are stored on Google Drive in the directory named "yolov3".
   Also keep on mind you would like to have the correct image format in the code:
   We used jpeg.



```
[ ]  import glob
     images_list = glob.glob("data/obj/*.jpeg")
     print(images_list)

[ ]  #Create training.txt file
     file = open("data/train.txt", "w")
     file.write("\n".join(images_list))
     file.close()
```

6. Train the model using YOLO based on the labeled dataset (The duration is about five hours if you have a big database).
7. If your train stopped, you can restart from last position if you comment and uncomment the commands below:



6) Start the training

```
# Start the training
!./darknet detector train data/obj.data cfg/yolov3_training.cfg darknet53.conv.74 -dont_show
# Uncomment below and comment above to re-start your training from last saved weights
# !./darknet detector train data/obj.data cfg/yolov3_training.cfg /mydrive/yolov3/yolov3_training_last.weights -dont_show
```

Notebook:
https://drive.google.com/drive/folders/1uyvMTTuP-sh4__OVTybHRw_nGBVougcs?usp=sharing

Detecting:

Extract Frames & Detecting, Extracting and Classifying Characters

An in depth description of our project:

*see note below.

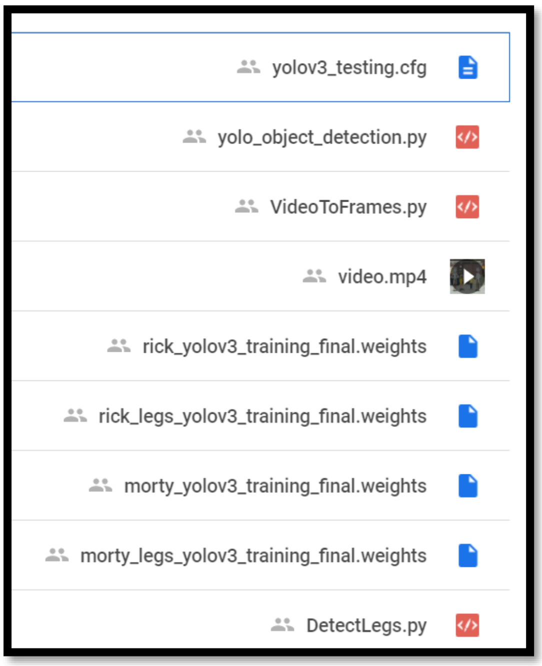
1. We implemented a function that requires a video named video.mp4 in your folder.
2. Input the number of frames that you desire.
   The code extracts that number of frames from the video and stores them in a new directory.
3. Then it uses the new directory of frames and our trained model for detect Rick and Morty characters in each frame.
   The detection is done and then draws a rectangle around the characters.
4. The next step is to use the rectangle coordinates for cropping the characters from each frame.
5. Finally, given the set of cropped characters images, the function uses another model for classifying their poses and determine if each character is standing or sitting.
6. The final results will be showed on the screen.

Running Instructions

Download the code, the input video, and our trained models to a single folder:

Python scripts:

https://drive.google.com/drive/folders/17JqHJkPIzuZISxj2u-cN9yJ9Os287zsU?usp=sharing



yolov3_testing.cfg
yolo_object_detection.py
VideoToFrames.py
video.mp4
rick_yolov3_training_final.weights
rick_legs_yolov3_training_final.weights
morty_yolov3_training_final.weights
morty_legs_yolov3_training_final.weights
DetectLegs.py

- **Then run this command in the CMD terminal in the correct path of the directory:**
  >python yolo_object_detection.py
  Notice: you will be asked to enter the number of frames you wish to extract from the given video (we recommend 20).
  Use the arrows on the keyboard for watching the output.

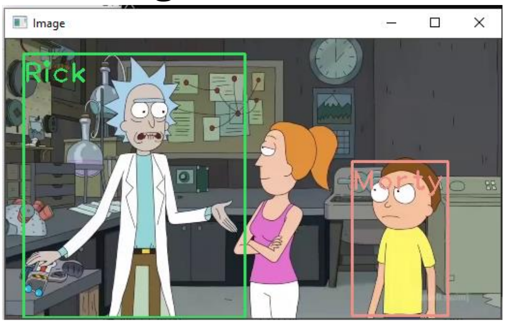
Important note:

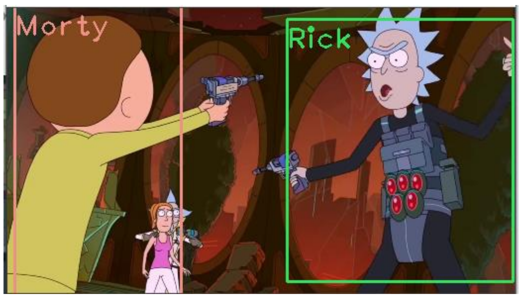
We used pycharm to run these codes.
In order to run the code you must have python 3.8, ANACONDA
Or be sure you have these packages installed:
opencv-python 4.4.5, cv2 4.4.2, numpy ,glob, random, os.

If you have problems with cv2:
Create in pycharm new anaconda virtual env 3.8, try give it permissions to all the packages.
If that doesn't work try this:
Go to File->Settings->Project Interpreter and then add by '+' button 'opencv-python' module on this repository.
It downloaded without an error for my PyCharm and also downloaded the dependencies as well.
https://stackoverflow.com/questions/51114826/cannot-import-cv2-on-pycharm

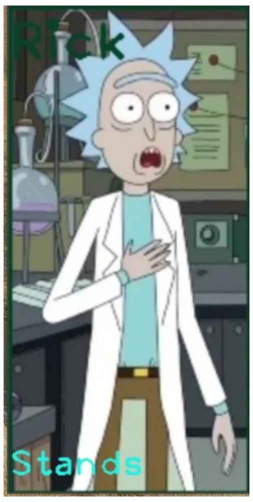
Output examples

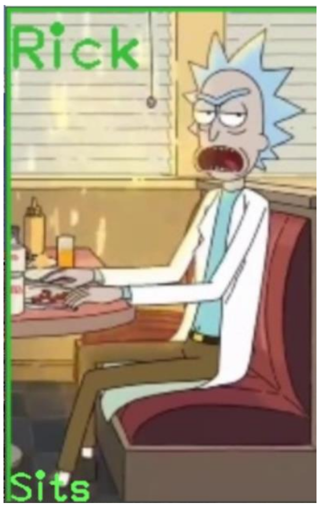
Detecting characters:



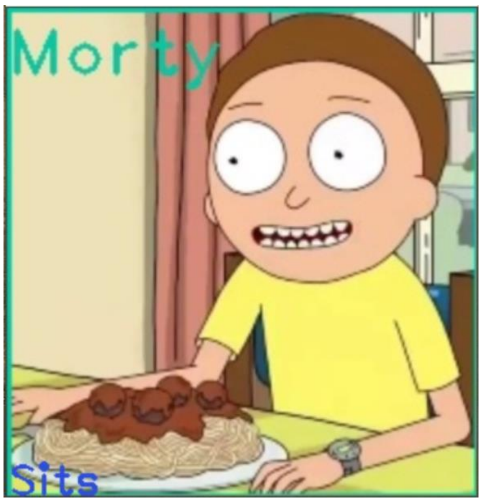


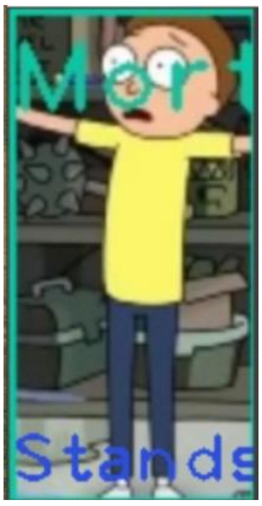
Rick:





Morty:

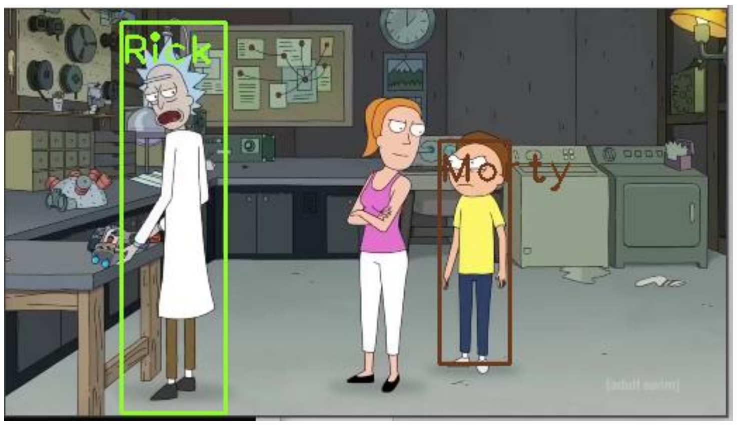


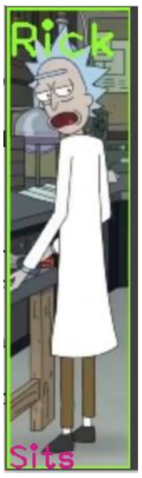


Few more words on the detecting and classifying:

In the detecting characters phase, we might fail to detect characters that stand toward us with the back, behind objects and with different than usual clothing. That might happen due to lack of training rare examples.
In the classifying phase (stands or sits), we might fail to classify characters that “stand” with the back or behind objects, or when we can see only upper body part.
We will classify these characters as “sits” due to lack of training of legs rare examples, or only upper body parts so there are no legs.

For example, in this image there is a rare legs position that was not trained on:





Summary

The results are very satisfying, and we have achieved our goal.
Our training models performed effectively, although relatively a small dataset (600 images for each model).

we succeeded in detecting and classifying Rick and Morty characters. Our models have over 90% success in detecting characters and about 70%-80% success in classifying their poses.