Image retrieval:

1. Firstly, make sure you are connected to the VPN and retrieve the captcha's list of filenames in the csv format. Then run the file *fetch.py*. In this file, change your "*shortname*" to your assigned TCD username and for the command "with open" command put your ("yourshortname" -challenge-filename.csv"). Then run the code.
2. After the generation of image files, make sure you save them in the folder of the project.

Generation:

1. Create a virtual environment in Python to use.
2. Install the required packages that are essential to run the project:
    a. captcha.image
    b. ImageCaptcha
    c. random
    d. os
    e. time
    f. request
    g. get
    h. ultralytics
    i. numpy
    j. pandas
    k. opencv-python
    l. tqdm
    m. matplotlib

Train/Val set generation:

For Train set:

1. run imageGen.py
    a. you will pass 5 arguments
        i. --*font1*, the path to the first font file
        ii. --*font2*, the path to the second font file')
        iii. --*output_dir*, directory to save generated CAPTCHA images
        iv. --*num_samples*, number of samples per character
        v. --*log_file*, path to the log file

For Val set :

1. run validateSet.py
    a. you will pass 3 arguments
        i. --*train_dir*, directory of training dataset
        ii. --*validation_dir*, directory of validation dataset
        iii. --*num_images*, num of images in the validation dataset

Preprocessing

After generating the data, we preprocessed the datasets separately. to preprocess the test data (4000 images from the csv file)

1. run *preprocess_test.py*
    a. you will pass 3 arguments
        i. --*image_dir,* path of captchas
        ii. --*save_dir,* path to store the preprocessed captchas
        iii. --*log_path,* to log the time.
2. run *preprocess_trainVal.py*
    a. you will pass the 3 arguments,
        i. --*image_dir,* path of captchas
        ii. --*save_dir,* path to store the preprocessed captchas
        iii. --*log_path,* to log the time.

Training

1. run the *train.py* file.
    a. you will pass 5 arguments
        i. --*model_path*, Path to the YOLO model weights file
        ii. --*data*, path to the data directory
        iii. --*epochs*, number of epochs for training
        iv. --*imgsz*, image size for training
        v. --*project*, project directory to save the model results

Export

1. run export.py
    a. you will pass 1 argument
        i. --*model_path*, path to the YOLO model weights file to export

Classify

1. run *classify.py*
    a. you will pass 3 arguments
        i. --*model_path*, path to the YOLO model weights file converted to onnx
        ii. --*input_dir,* path to the directory containing segmented test images
        iii. --*output_csv*, path to save the output CSV file