Anomaly Detection in Surveillance Videos

This document contains explanations and steps for executing the codes related to the project on 'Anomaly Detection in Surveillance Videos'. It explains the directory structure, training, testing and the summarization steps.

Directory structure:

1. *video_root_path* – must be the root directory for your datasets
2. Create a folder corresponding to your dataset inside the *video_root_path* directory. Inside the folder, place the training and testing videos inside *training_videos* and *testing_videos* folders.
3. Also create folders named *training_frames* and *testing_frames* inside the dataset folder. This is where the frame extraction program will store the extracted frames.
4. Additionally, with the available datasets, groundtruth, reconstruction-frame plots and rating files can be found under *groundtruth*, *graph_plot*, *rating_training* and *rating_testing* folders. For new datasets, these folders should be created manually.
5. *data_preprocessing.py* should be placed along with the training code in the same directory.

Training:

1. *unsupervised_training.py*
    - Command: **python3 unsupervised_training.py**
    - Code for training the unsupervised autoencoder.
    - Change the value of the variables *dataset*, *batch_size*, *time_length* and *iterations* according to the requirements.
    - *Comment prep.normalize_input() [line 128] and prep.create_input() [line 131],* if data preprocessing is not required. Sync the value of *time_length* in the preprocessing file with the value in the training file.
2. *supervised_training.py*
    - Command: **python3 supervised_training.py**
    - Code for training the supervised autoencoder.
    - Same as above, for changes and comments

Testing:

1. *testing.py*
    - Command: **python3 testing.py > output.txt**
      here, output file will store the reconstruction error per frame.
    - Ensure that the values of *time_length*, *batch_size* and *dataset* are same as the values during training.
    - Output is only the reconstruction error of all the frames of all the test videos. In order to execute the script evaluation and creation of summaries, manually create files corresponding to the video names and copy-paste the reconstruction errors. Check the path where the individual files (check in the summary creation script)
2. *testing_Grad_CAM.py*
    - Command: **python3 testing_Grad_CAM.py > output.txt**
    - Ensure the values of the variable as mentioned earlier.
    - Output is a tuple <reconstruction error, frame number> for each video of the test set. For summary creation manually create files corresponding to the video names and copy-paste the reconstruction errors. Check the path where the individual files (check in the summary creation script)

Anomaly Summarization:

1. *thresholding.py*
    - Command: **python3 thresholding.py <video_number>**
    - Uses the thresholding mechanism, described in the paper, for creating summaries.
    - For creating summaries of the entire test dataset, use the helper script *generate_result.sh.* For executing the shell script, write the following command: *./generate_result.sh <number of the last video>*
    - As output, we get a summary video corresponding to the video number
2. *sorting_selection.py*
    - Command: **python3 sorting_selection.py <video_number>**
    - Uses the sorting and selection mechanism for creating summaries.
    - For creating summaries of the entire test dataset, use the helper script *generate_result.sh.* For executing the shell script, write the following command: *./generate_result.sh <number of the last video>*

- *As output, we get a summary video corresponding to the video number*

3. *sliding_window.py*
    - Command: **python3 sliding_window.py <video_number>**
    - Slides a window across the duration of the entire video and picks that segment which has the maximum reconstruction error.
    - For creating summaries of the entire test dataset, use the helper script *generate_result.sh.* For executing the shell script, write the following command: *./generate_result.sh <number of the last video>*
    - *As output, we get a summary video corresponding to the video number*
4. *generate_result.sh*
    - Command: **./generate_result.sh <number_of_videos>**
    - Runs a loop from 1 to n, where n is the total number of testing videos. So, ensure that the videos are numbered from 1 to n.
    - Depending on the kind of summary required, change the file name inside this file. 2 commands have been commented for illustration sake.

Helper Scripts:

1. *parse_json.py*
    - Command: **python3 parse_json.py**
    - Takes a json file, which has been output after rating segments using oTranscribe tool and produces groundtruth/rating files in text format.
    - Change the input file in the program manually and place the input json file in the same directory.
    - Update the value of *factor, fps* and *dataset* as required.
2. *extract_frames.sh*
    - Command: **./extract_frames.sh**
    - Change the value of the variable corresponding to *dataset*, *training_video_folder* and *testing_video_folder* according to your requirements.
    - Contains extraction of all the frames and saving them as both *.jpg* and *.png.* Comment relevant portion, if not required.
3. Few other scripts have also been published. These are not directly related to this work, but can be useful for future endeavours.

- *extract_darknet_features.sh* extracts darknet features of each frame of all the videos. Output is a file for each frame. These files are combined to form a single feature file for each video using the program *proprocess_after_darknet.py*
- *preprocess_after_darknet.py* combines the files containing the extracted features and creates a combined feature file for each video.