SUSPICIOUS VEHICLE DETECTION

Luis Ivey, Paul Stephen Hutchinson Maltaghati

Boston University
Department of Electrical and Computer Engineering
8 Saint Mary's Street
Boston, MA 02215
`www.bu.edu/ece`

May. 06, 2022

Technical Report No. ECE-2022-2

Summary

With the rise of computer vision techniques, today's video surveillance has become a sophisticated means of deterring crime, increasing safety, and providing evidence of criminal activity. However, the surveillance of parked suspicious vehicles and the automatic recognition of the activities involving these vehicles are still an ongoing exploration that current video surveillance systems fail to offer. This project aims to combine existing methods as much as possible to demonstrate the potential of modern image/video processing and computer vision methods for detecting suspicious vehicles. The system design was determined after reading and understanding video processing and deep learning techniques researched and described in academic literature. The project attempts to accomplish both object detection and action recognition. The detection of suspicious vehicles are based on video recordings taken from the Photonics cameras in a region of interest in front of Marsh Plaza. The action recognition aspect of the code will determines the interactions people are having with the vehicle.

Contents

List of Figures

List of Tables

1 Introduction

In recent years, object detection has been one of the most challenging yet essential computer vision tasks. The localization and classification of visual objects in digital images provide a strong foundation for other computer vision hurdles, such as object tracking, image segmentation, image captioning, facial recognition, and much more [1]. Object detectors are also the backbone of most intelligent surveillance systems [2]. However, today's intelligent surveillance systems are only equipped for specific motion detection and object tracking tasks and cannot be used to draw semantics from the scene [2]. Therefore, the underlying goal of these systems, such as deterring crime, increasing safety, and providing evidence of criminal activity, are less reliable because they cannot conclude when such events are happening. For example, the detection of suspicious vehicles.

Vehicles such as those parked carelessly, abandoned while still running, and or with the doors left open are considered suspicious. Some other scenarios, such as loading or unloading vehicles, are questionable, especially during unusual hours. Nevertheless, intelligent surveillance systems would only be able to detect and record all vehicles on the scene. The reviewing process would then be conducted after the fact by a human to determine if a vehicle was indeed suspicious or not. These systems lead to inefficiency because of the extra step in reviewing the video feed offline. Action recognition would need to be introduced to automate the entire process and allow the intelligent surveillance system to draw its conclusions from the scene.

However, recognizing actions in videos is challenging because of the temporal component and what information can be deduced based on motion between frames. Most research communities have concluded that a two-stream process that separates a video's temporal and spatial layers shows promising action recognition results. Object tracking must also be leveraged to reduce the region of interest of the scene and reduce the number of frames being fed to the computational intensive action recognition pipeline. The fusion of object tracking derived from object detection and action recognition methods would produce a more reliable and efficient system removing the need for offline reviewing because of automatic recognition of specified events.

2 Literature Review

2.1 Two-Stage and One-Stage Object Detection

Object detectors solved by deep learning methods can be classified into two categorize two-stage and one-stage detectors. Two-stage detectors, for example, R-CNN, were one of the first object detectors to use deep learning as a solution [3]. However, R-CNN and other two-stage detectors are significantly slower and cannot be used in real-time solutions. One-stage object detection frameworks were created to solve the real-time practicalities that two-stage detectors could not. The main difference between two-stage and one-stage object detectors is that one-stage detectors is an end to end computationally low-costing neural network being applied to the image. Figure 1. illustrates the end to end of two popular one-stage pipeline. Popular object detectors are Faster R-CNN [4], Single Shot Detectors (SSD) [5], and You Only Look Once (YOLO) [6].

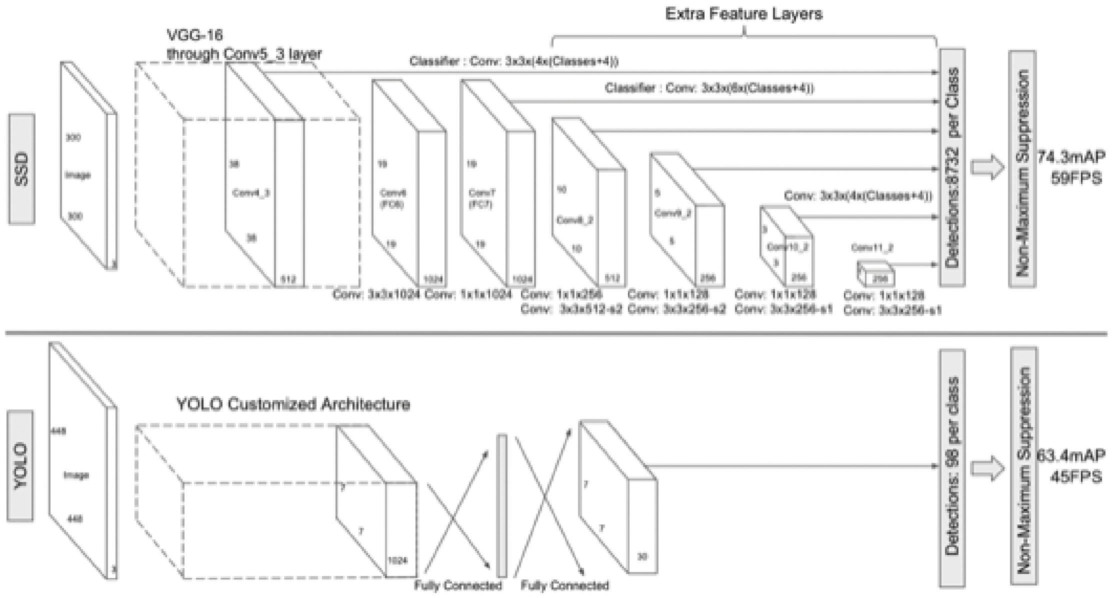
Extra Feature Layers
VGG-16 through Conv5_3 layer
Classifier : Conv: 3x3x(4x(Classes+4))
Classifier : Conv: 3x3x(6x(Classes+4))
Conv: 3x3x(4x(Classes+4))
SSD
Image
Conv4_3
Conv6 (FC6)
Conv7 (FC7)
Conv8_2
Conv9_2
Conv10_2
Conv11_2
Conv: 3x3x1024
Conv: 1x1x1024
Conv: 1x1x256 Conv: 3x3x512-s2
Conv: 1x1x128 Conv: 3x3x256-s2
Conv: 1x1x128 Conv: 3x3x256-s1
Conv: 1x1x128 Conv: 3x3x256-s1
Detections:8732 per Class
Non-Maximum Suppression
74.3mAP 59FPS
YOLO
YOLO Customized Architecture
Fully Connected
Fully Connected
Detections: 98 per class
Non-Maximum Suppression
63.4mAP 45FPS

Figure 1: SSDs and YOLOs convolutional neural network architectures [5]

Faster R-CNN was the amalgamation of previous object detection R-CNN family of frameworks which introduced region-based Convolutional Neural Networks (CNNs). The computational cost of region-based CNNs has drastically reduced from [3] by using convolutional methods from SPPnet [7] and Fast R-CNN [8]. These methods include computing the feature maps from the image and then pooling features into arbitrary regions or grouping, which avoids repeatedly computing the convolutional features. The Selective Search algorithm is used to select the regions which are demonstrated in Figure 1. However, these methods still produce a bottleneck in their region voting systems. Faster R-CNN solves the time spent on region proposals by an algorithmic change that introduces nearly cost-free Region Proposal Networks (RPNs). The RPNs break away from the Selective Search methods and rely on speeding up the R-CNN framework by sharing computation among convolutional layers and sliding neural networks to propose regions instead of selective search. Yet, Faster R-CNN is still considered a two-stage detector because of a second iteration through the R-CNN after the regions have been selected. Using a VGG-16 model for classification, seven frames per second frame rate on the Pascal VOC 2007 was achieved.

From The University of North Carolina at Chapel Hill, Wei Liu proposed a new neural network architecture that eliminates the two-step method of a proposal generation system and features resampling stages into one single network [5]. SSDs training

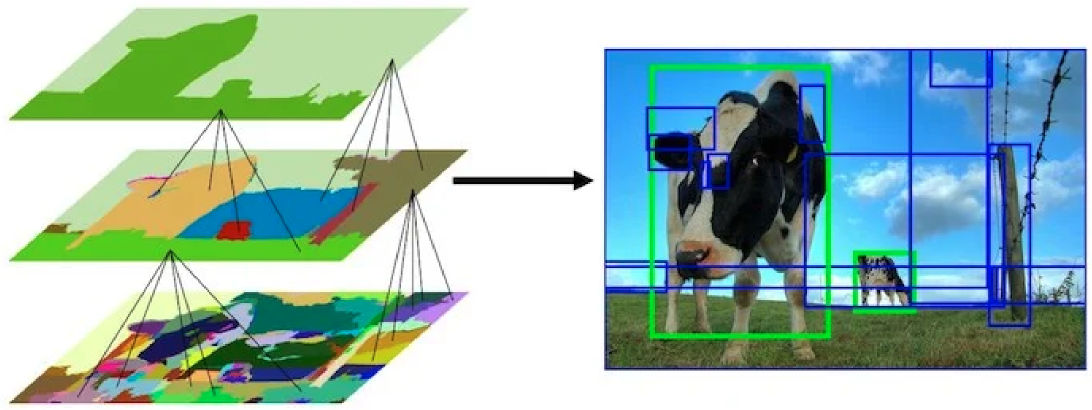

Figure 2: Demonstrates the Selective Search process.

process consists of an input image that is divided into a grid. A small set of default boxes (or anchor boxes) predicts each given object's probability within the boxes, shown in figure 2. The default boxes also have different aspect ratios that boost

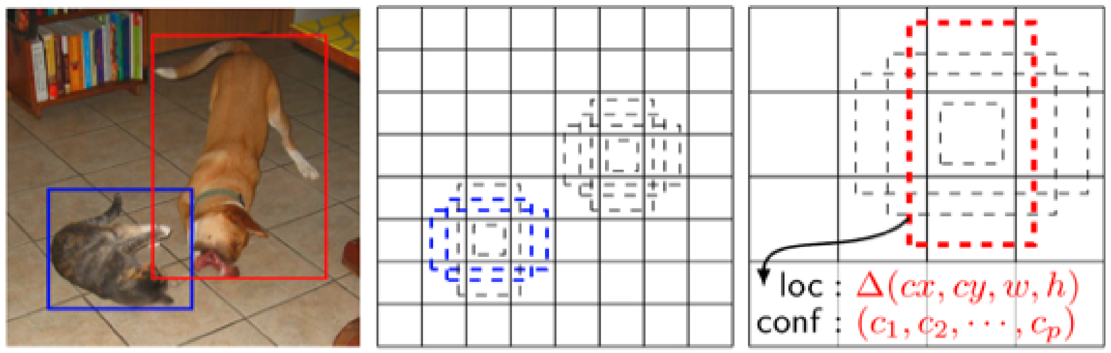
loc : $\Delta(cx, cy, w, h)$
conf : $(c_1, c_2, \cdots, c_p)$

Figure 3: SSD: Single Shot MultiBox Detector framework [5]

detection speed and accuracy when applied to later stages of the network. The deep learning model's loss uses Smooth L1 and Softmax for a weighted sum between its localization loss and confidence loss. However, YOLO and its subsequent variations are often utilized because of their ease of use, faster speeds because of YOLO's predictive boxes instead of iterative anchor boxes, and open-source deployable packaging compared to SSDs.

YOLO, a method proposed by Joseph Redmon from The University of Washington, takes a similar approach to [3] for region voting. The entire image and divides it into a grid-based voting system of potential bounding boxes, which each grid predicts N bounding coordinates and confidence using convolutional features. However, a spatial constraint is in place to reduce the number of detections for the same object. YOLO learning procedure can be summed up into three steps: resizing an input image, running the resized image through a single convolutional neural network, and reducing predicted bounding boxes. Ninety-eight bounding boxes per image compared to Selective Searches' two thousand bounding boxes. The simultaneous predictions of multiple bounding boxes compared to the iterative process of the sliding windows method increases the speed of detection and achieves a larger spatial context which leads to fewer mistakes.

2.2 Action Recognition

Current general action recognition solutions start with a two-stream convolutional neural network (CNN). Karen Simonyan and Andrew Sizzerman, part of Oxford's visual geometry group, were the first to propose the two-stream CNN based on the human visual cortex and how humans perceive motion [9]. Their CNN architecture mimics the ventral (used for object detection) and dorsal (recognition of activity) stream of the human eye by keeping the task of action recognition decoupled into spatial and temporal ConvNets. Figure 4 shows the two-stream ConvNets. The

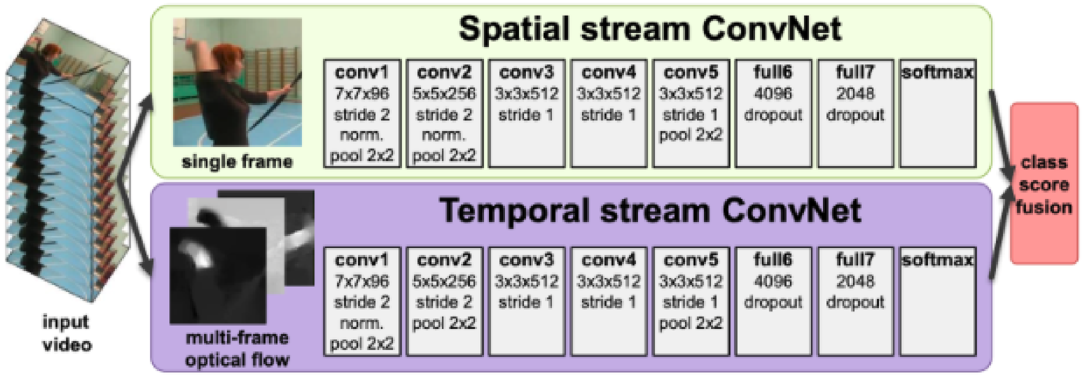

Figure 4: The decoupled two-stream ConvNet architecture [9].

inputs for the spatial network are frames of a short video within a more extensive video where the action takes place. The inputs for the temporal network are from the same consecutive frame's optical flow displacement fields stacked together. However, the two-stream architecture does not incorporate the entire video, and therefore the long-term coherence of the temporal structure is missing.

Rohit Girdhar and the Robotics Institute members at Carnegie Mellon University introduced an extended version of a 2D metric learning Vector of Locally Aggregated Descriptors (VLAD) [10]. Their solution is an end-to-end feature descriptor aggregator that takes the entire video and uses a modified version of the NetVLAD aggregation layer named Action VLAD. The authors explored different aggregation locations and investigated how to combine the signals, the flow, and RGB streams. Figure 5 shows the ActionVLAD layer. Although reaching accuracies of 93.6 on the standardized dataset UCF101, months of tuning are unavoidable to achieve a good feature representation when training 3D ConvNets.

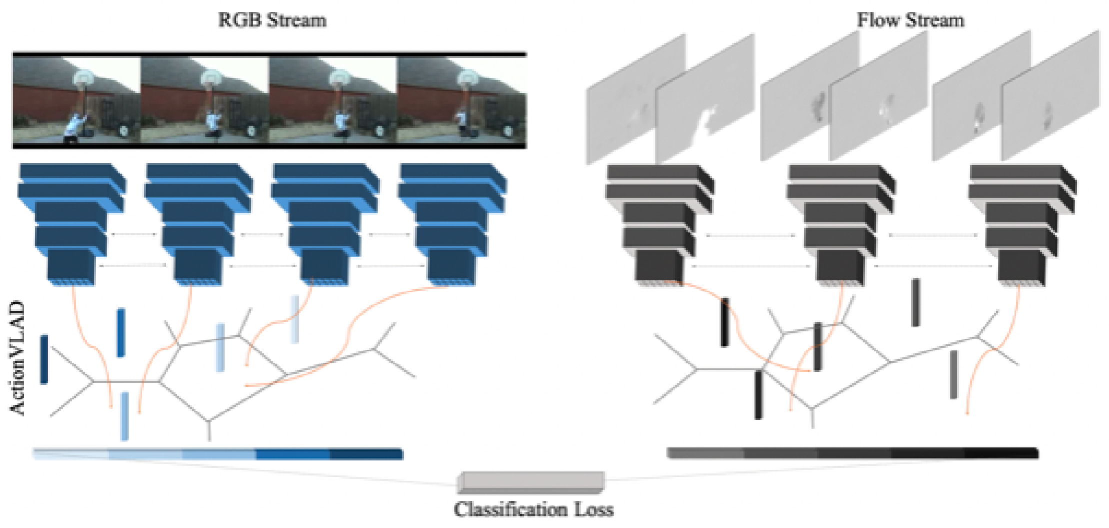

Figure 5: ActionVLAD's aggregation layer [10].

As a solution to the initial gathering of proper feature representation when first training 3D ConvNets from scratch, the Temporal 3D ConvNets: New Architecture and Transfer Learning for Video Classification proposes a transfer learning approach to initialize a 3D CNN randomly [11]. The authors have also used their architecture to efficiently capture an entire video's appearances and temporal information without optical flow maps. The process is done by first using the supervision transfer of a 2D ConvNet onto a 3D ConvNet to learn the spatial parameters effectively. Figure 6 shows the architecture for transferring knowledge between the 2D and 3D ConvNet. A novel layer named Temporal Transition Layer (TTL) is introduced to concatenate the temporal feature maps extracted at different temporal depth ranges.

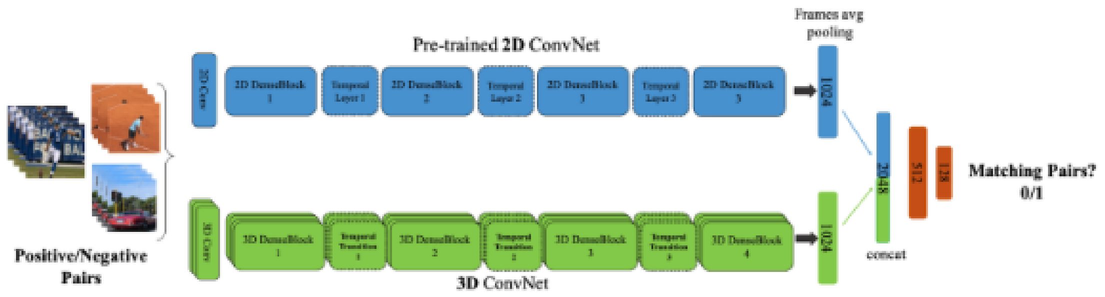

Figure 6: The architecture for transferring knowledge between the 2D and 3D ConvNet [11].

Yet another approach to general action recognition is the various skeleton-based

ConvNets. Raw Skelton points and the motion of these points are fed into CNN for label prediction [12]. However, many of these methods can not deal with irregular skeleton joints when connecting consecutive frames. A proposed graph regression-based Graph convolutional network (GCN) aims to capture the Spatio-temporal variation of skeleton point data [13]. Another way of optimizing a skeleton-based approach is to use rolling maps from 3D skeleton data [14]. The 3D geometrical connection between the body points provides a significant description of the human skeleton in action recognition instead of absolute locations. Therefore, a 3D rotation between different body parts has been used to improve the accuracy of action recognition.

3 Problem Statement

This project aims to combine existing methods as much as possible to demonstrate the potential of modern image/video processing and computer vision methods for detecting suspicious vehicles. The computer vision techniques used in this project are data-driven with machine learning techniques to (1) detect and classify (e.g., passenger car, truck, van) a parked/stopped motor vehicle in a selected area (e.g., no-stopping lane on a Common-wealth Avenue or its sidewalk), (2) detect the presence of people interacting with the vehicle and identify what they are doing (e.g., getting into/out of a car, loading/unloading luggage, throwing trash bags onto a truck's flatbed). The project's desired outcome is to capture videos from Commonwealth Avenue and demonstrate the effectiveness of the project's approach offline on those captured videos. The offline results are assurance that a real-time application that can capture video frames directly from one of the Photonics cameras (e.g., using ffmpeg), detects/classifies vehicles in restricted areas, and identifies people's actions is entirely possible.

4 Implementation

After carefully reviewing the various papers on Action Recognition and one-stage object detectors, an understanding of the steps and final solutions was concluded. A data-driven approach was taken by using the Photonics rooftop cameras. Due to the strict turnaround time for this project and the priority of having a run-able project, off-the-shelf solutions were leveraged to speed up the project's production time. The scope of the project is also reduced by assuming a specific camera angle of Marsh Plaza and that stationary vehicles will most likely be parked on the side of the road. Figure 7 illustrates the pipeline of the project during run-time. The main driver will load the video clip, create a instance of our scene class, calculate the timestamps for each frame, and load the YOLOv5 model. The main driver will then pass the frame, timestamp, and frame number to the Scene class. The Scene class holds most of the algorithms that will simultaneously handle the detection of stationary vehicles, the classification of those stationary vehicles, and the highlighting of objects near the vehicle by annotating the frame whenever these events happen.

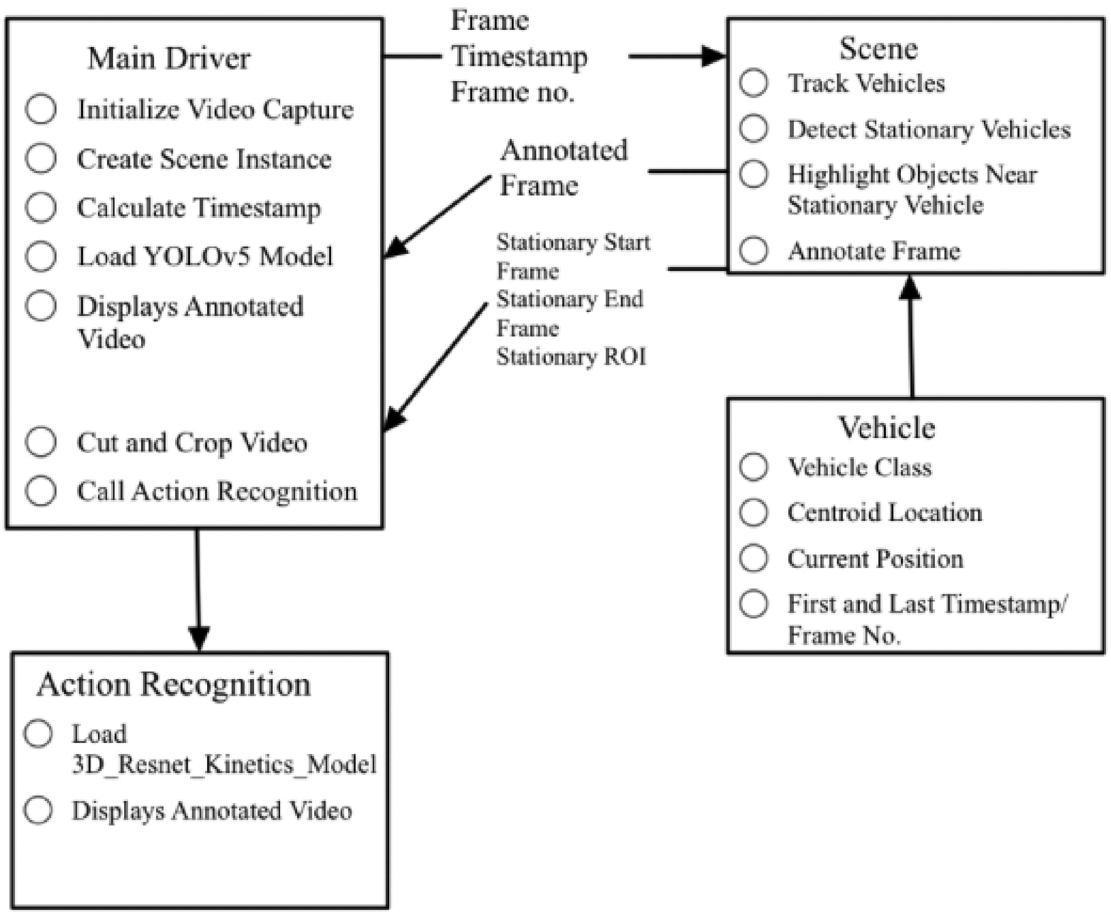

Figure 7: Block diagram of the project.

The first task for the Scene class is to detect vehicles in the scene. Stationary vehicles are found by feeding each frame to the YOLO model. YOLOv5 will be used for this project which is the successor to YOLOv4 [15]. The model is provided by Glenn Jocher, who introduced YOLOv5 with PyTorch support found here https://github.com/ultralytics/yolov5. However, the YOLOv5 model that the project is using was trained on the MS COCO dataset [16]. The MS COCO dataset includes the labels of many different vehicles, eliminating the need to retrain the YOLOv5 model with transfer learning techniques. As a result of feeding the frame to the model, the inference stage returns four points (two starting points and two ending points) for each detected object representing its location within the image. The sifting process then cycles through all the predicted objects and locates only the predictions labeled as one of the vehicle classes from the MS COCO list. To reduce the number of vehicles being tracked, a mask was applied to the original image that includes a small segment of the street in front of Marsh Plaza and sets the rest to black pixel intensity shown in Figure 8.

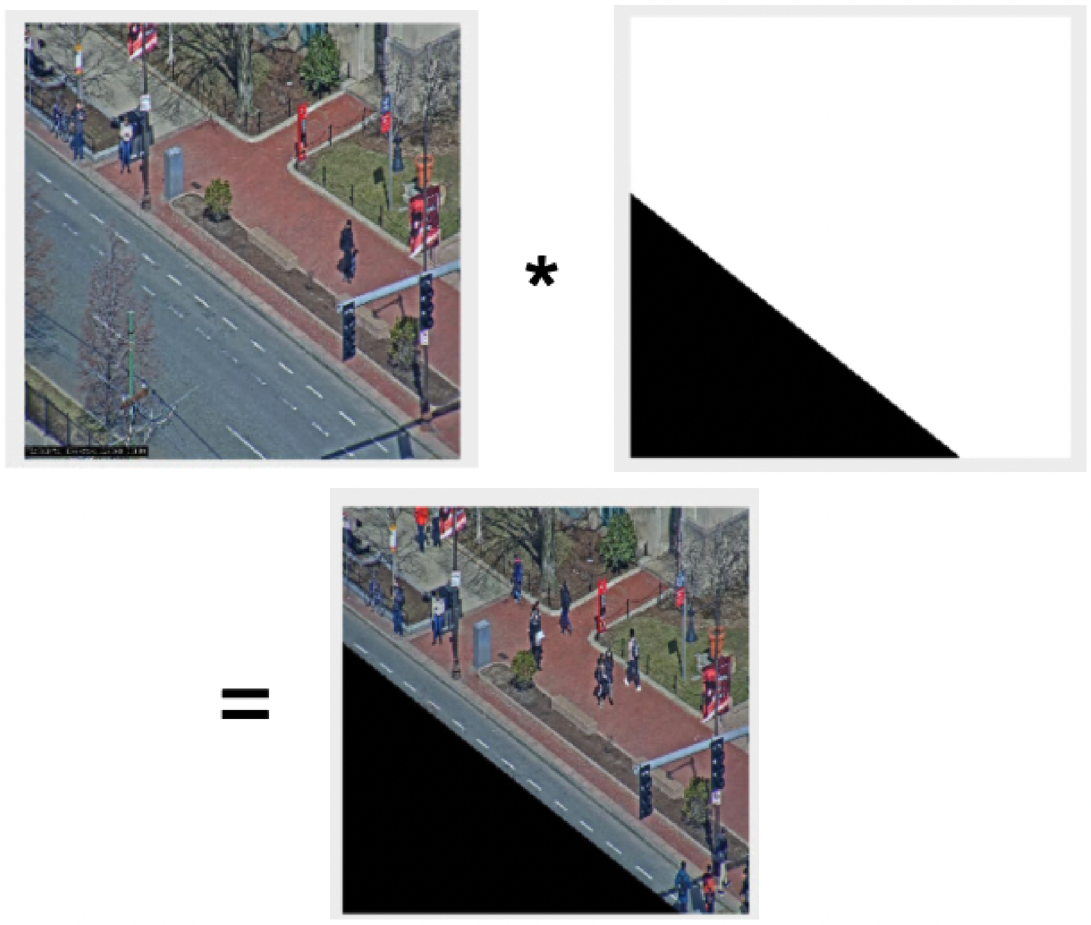

Figure 8: A copy of the original frame is masked for the inference step.

Only vehicles in this region are being checked to see if they are stationary. By only checking in this region, a reduction in the number of vehicles being tracked is a necessity keeping the real time goal of the project. Yet, another method of speeding up the project was to reduce the resolution size of 1920x1080 to 300x300 to increase inference speed. A ratio was taken to relate the bounding boxes in 300x300 to the 1920x1080 resolution (shown in Figure 9). Another pass of the original image to the model was done (without a mask being applied) to obtain the location of the rest of the vehicles for the annotation process of drawing the bounding box. Once again, only vehicles that were in the specified region of interest are saved to a list. The vehicles from that list along with their bounding points, timestamp, frame number, and class label are passed to the vehicle tracker function for further evaluation.

The vehicle tracker function iteratively looks at all the vehicles that appear in the passed in list and checks their position to see if they have previously been detected. This is done by calculating the newly detected vehicles' center point based on the

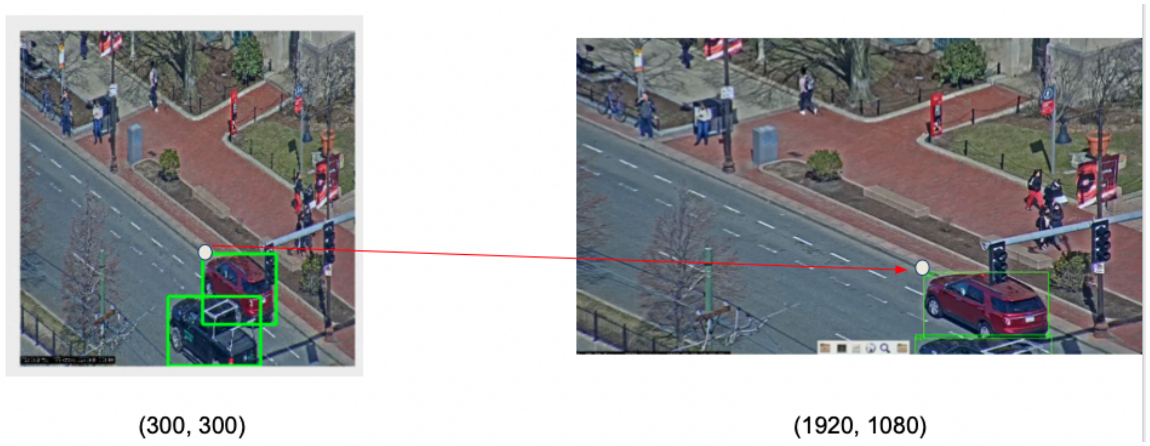

Figure 9: Example of resolution mapping

passed bounding points and loops through the Previously Detected Vehicles Class List (PDVCL). The difference in center points is calculated and compared to a predefined threshold. If the threshold is exceeded, the new vehicle will be inserted into the PDVCL. If the difference in center points is below the threshold, the vehicle in the PDVCL position will be updated. The vehicle class contains a number of previously located center points, and so the updating process will be done in a round-robin format. The vehicle tracker function will then ask each vehicle in the PDVCL whether or not it is stationary. The vehicle class reads its previous locations points and checks to see if they are the same. By empirically evaluating the center points and establishing a threshold for when a vehicle is stationary, the threshold of having all the points be the same produced the best results in detecting when a vehicle was stationary. The vehicle tracker function sets a stationary vehicle flag by placing the vehicle in a Stationary Vehicle List (SVL) based on the answer from the vehicle class. The position of the vehicles in the SVL are used to detect objects around the vehicle by expanding the vehicles top left point and bottom right point by ten and using that new region of interest to sift through object only in that location for the following frames. New objects around the vehicle are then highlighted in the frame annotation step. With the vehicles in the SVL, the subsequent frame will have a visible box around the stationary vehicle and any object around the vehicle from a list of labels. Once there is no detection's with in the region of interest for two consecutive frames and the stationary vehicle that raised the stationary flag is no longer stationary, the current time and frame number is saved to be used in clipping function.

The video clipping occurs within a separate function which is called within the driver. The driver passes the original video which has to be clipped, the location of the suspicious region of interest within the original video, and the frames which the new video will start and stop on. In the cropping function, video processing is done using Open CV. First, an Open CV VideoWriter method in order to create a writer object. This establishes the .avi file which we want to store the new video in. Then, as each frame of the original video is looped over, the code checks to see if the frame number is the same as the start frame number which was fed as an input. When it is determined that the suspicious scene has begun, the scene is resized to only included the region of interest, and the frames are written to the .avi file. The .avi file is saved within the same workspace as the rest of the project.

The final step of the project is action recognition. The action recognition is the last method called within the main driver, and it utilizes an imported data set and residual neural network (RNN). The action recognition first calls the data set label text file and the RNN module which are both stored in the workspace. The machine learning module is read in using the Open CV Deep Neural Network (dnn) method. Open CV methods then allow for a blob of frames created. These frames are then fed to be the input for the loaded RNN module. The module then outputs one of the 400 actions which it has been trained to recognize. The predicted action is then added as a text box onto the video as it loops over more frames. Since the code is running in real-time, the RNN continuously updates, and since lots of events are occurring within the region of interest, the predicted action changes multiple times. This is due to the fact that the main actions occurring within the video, such as getting out of a car, cannot be detected by the kinetics data set. These results are further addressed within the conclusion of the paper.

5 Experimental Results

Table 1: SVD Performance in Detecting Stationary Vehicles

Clip No.	**Vehicle**	**Action**	**Stationary Detected**
1	Red SUV	Dropoff	Yes
2	White SUV	Dropoff	Yes
3	Gray Sedan	Dropoff	Yes
4	Gray SUV	Dropoff	No
5	Blue SUV	Dropoff	Yes
6	Black SUV	Walking	No

Table 2: SVD Performance in Vehicle Classification

Clip No.	**Vehicle**	**Action**	**Vehicle Classified**
1	Red SUV	Dropoff	Yes
2	White SUV	Dropoff	Yes
3	Gray Sedan	Dropoff	Yes
4	Gray SUV	Dropoff	Yes
5	Blue SUV	Dropoff	Yes
6	Black SUV	Walking	Yes

Table 3: SVD Performance in Action Recognition

Clip No.	**Vehicle**	**Action**	**Action Classified**
1	Red SUV	Dropoff	Yes
2	White SUV	Dropoff	Yes
3	Gray Sedan	Dropoff	Yes
4	Gray SUV	Dropoff	No
5	Blue SUV	Dropoff	Yes
6	Black SUV	Walking	No

Below are several resulting images from our project and the tested clip results can be found here https://drive.google.com/drive/folders/
14bd6ANYWuOdhQsplXOMVP1QmoO9Oa1gC?usp=sharing,

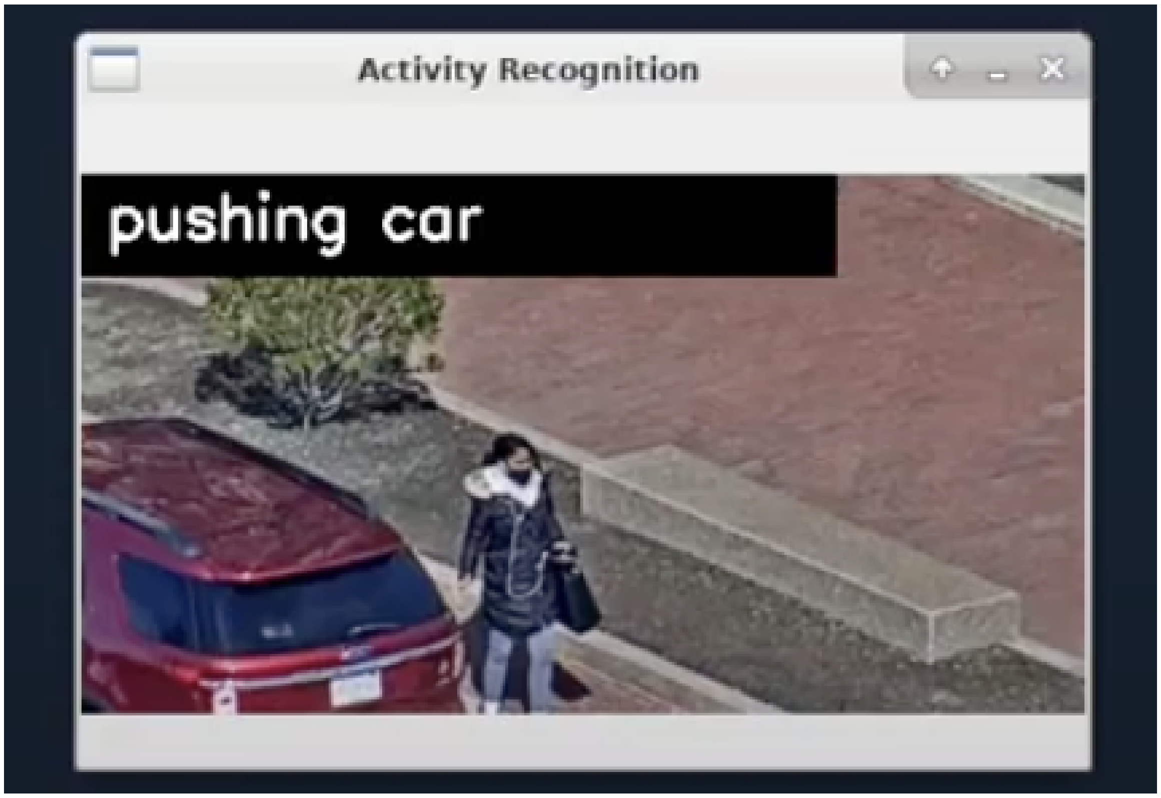

Figure 10: Action Recognition Results

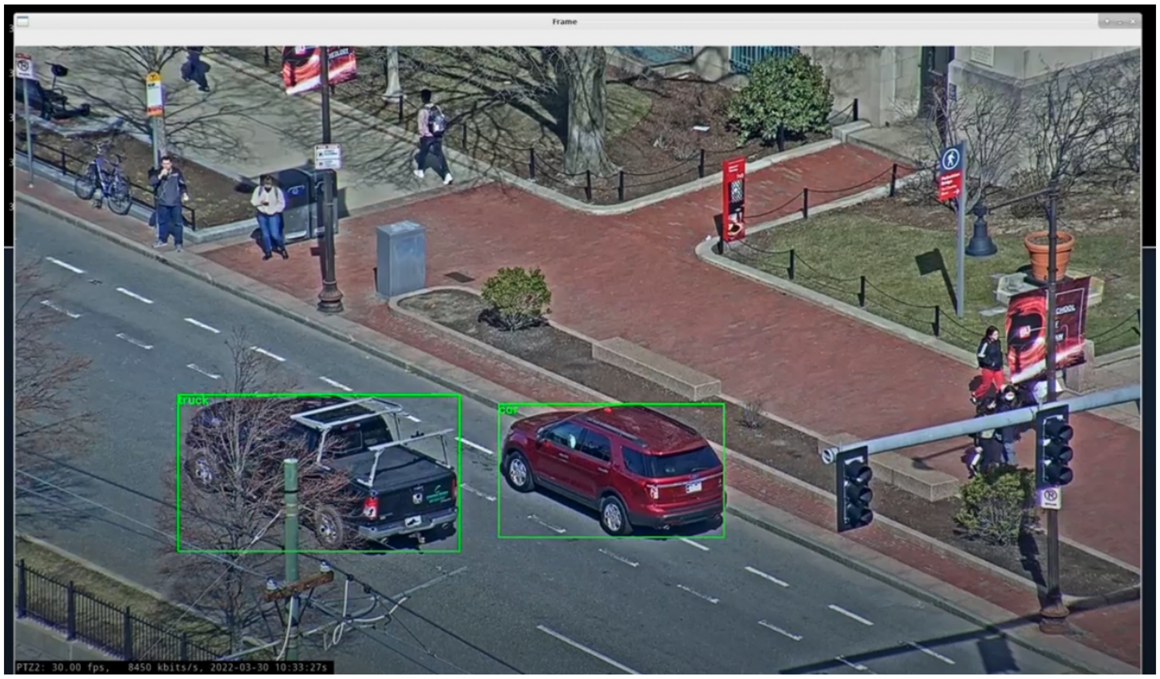

Figure 11: Red SUV Vehicle Tracking

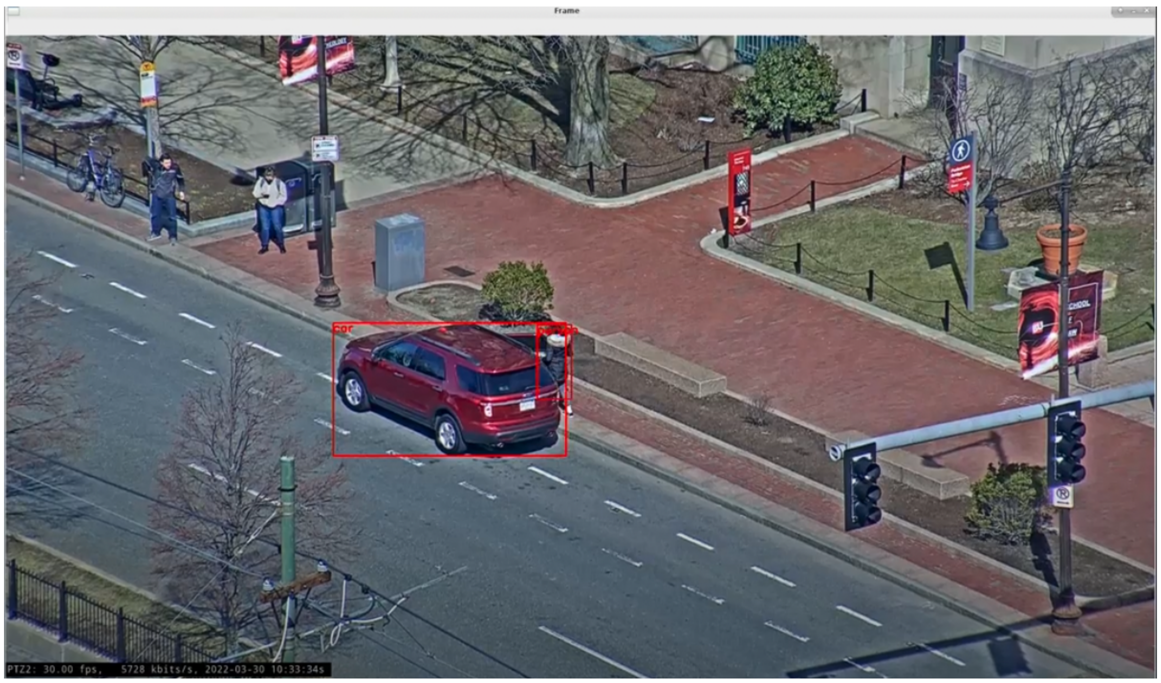

Figure 12: Red SUV Vehicle Parked

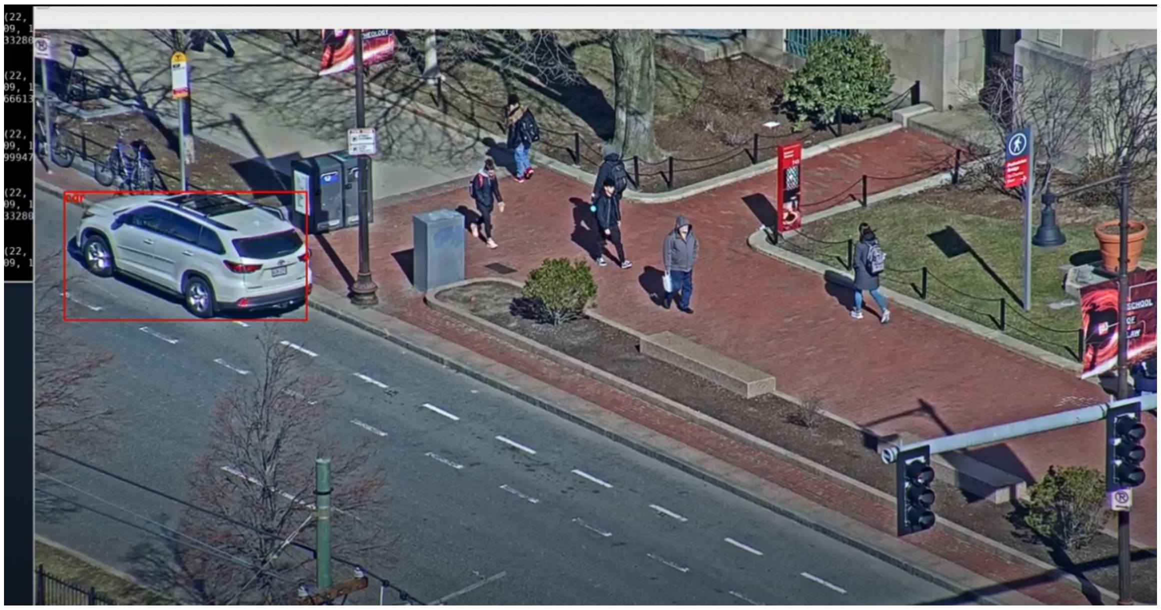

Figure 13: White SUV Vehicle Parked

The stationary vehicle detection, vehicle classification, and action recognition experimental results for our six test videos are displayed in the above tables. In every video, the YOLO model was able to classify vehicles as either cars or trucks. The high success rate of the classification is due to the large amount of training data in the COCO data set. There was high accuracy in our stationary vehicle detection, as four our of six stationary cars were detected. The other two stationary vehicles were not detected due to error which was outside of our control. In the fourth video, it can be seen that the wind was very strong. As a result of the strong wind, the Photonics camera shakes while the stationary vehicle is stopped on the side of the road. Since camera is continuously moving, the center of the stationary vehicle also moves with respect to the camera, and thus our code thinks that the whole vehicle is moving. In the sixth video, the stationary vehicle cannot be detected since half of it is outside of the camera's field of view. The action recognition does not work for these two videos because the action recognition function requires a cropped video to be an input. This cropped video is based off of the coordinates of a stationary vehicle's region of interest. If no stationary vehicle is detected, no cropped video will be created and saved to the workspace.

6 Conclusions

This project was successful in detecting vehicles and people, and it was also successful in filtering out suspicious vehicles from other vehicles on the street. From the results generated, it is clear that YOLO object detection works very well in the detection and classification of suspicious vehicles. As newer YOLO models continue to be released, the detection of suspicious vehicles will only improve. Going forward, work can be done in terms of making sure that vehicles can be detected regardless of the camera and the camera position. All of the data for this project was recorded while the camera was at the same position, and it is important to test the code with different devices and scenes. Another improvement which can be made for future work involves camera stability. The camera which was used to record video was installed in a way that allowed for wind to shake it. For the best results to be generated, a steadier camera would be ideal.

The implementation of action recognition was also a success in this project. The code was able to successfully crop the video based on the region determined by YOLO, and then it was able to successfully create a new video of the suspicious scene. The code also successfully called on the action recognition machine learning module. The main changes which can be done to improve the action recognition must be done in the kinetics data set and neural network themselves. The data set which was used only contains 400 actions. The more recent version of the data set contains 700 actions, but in the literature, it is stated that actions such as getting out of a car and getting into a car are unrecognizable due to a lack of data. It is clear, though, that researches are hoping to be able to detect actions which concern human interaction with vehicles, so within the near future it can be assumed that our code will be able to identity actions which will be of interest for applications ranging from surveillance to monitoring.

7 References

[1] Z. Zou, Z. Shi, Y. Guo, and J. Ye, “Object detection in 20 years: A survey,” arXiv.org, 16-May-2019. [Online]. Available: https://arxiv.org/abs/1905.05055.
[2] A. I. Adrian, P. Ismet and P. Petru, ”An overview of intelligent surveillance systems development,” 2018 International Symposium on Electronics and Telecommunications (ISETC), 2018, pp. 1-6, doi: 10.1109/ISETC.2018.8584003.
[3] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” arXiv.org, 22-Oct-2014. [Online]. Available: https://arxiv.org/abs/1311.2524.
[4] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards real-time object detection with region proposal networks,” arXiv.org, 06-Jan-2016. [Online]. Available: https://arxiv.org/abs/1506.01497.
[5] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, “SSD: Single shot multibox detector,” – arXiv Vanity. [Online]. Available: https://www.arxiv-vanity.com/papers/1512.02325/.
[6] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” You Only Look Once: Unified, Real-Time Object Detection – arXiv Vanity. [Online]. Available: https://www.arxiv-vanity.com/papers/1506.02640/.
[7] K. He, X. Zhang, S. Ren, and J. Sun, “Spatial pyramid pooling in deep convolutional networks for visual recognition,” arXiv.org, 23-Apr-2015. [Online]. Available:

https://arxiv.org/abs/1406.4729.
[8] R. Girshick, "Fast R-CNN," arXiv.org, 27-Sep-2015. [Online].
Available: https://arxiv.org/abs/1504.08083.
[9] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," arXiv.org, 12-Nov-2014. [Online].
Available: https://arxiv.org/abs/1406.2199.
[10] R. Girdhar, D. Ramanan, A. Gupta, J. Sivic, and B. Russell, "ActionVLAD: Learning spatio-temporal aggregation for Action Classification," arXiv.org, 10-Apr-2017. [Online]. Available: https://arxiv.org/abs/1704.02895.
[11] A. Diba, M. Fayyaz, V. Sharma, A. H. Karami, M. M. Arzani, R. Yousefzadeh, and L. Van Gool, "Temporal 3D convnets: New Architecture and Transfer Learning for Video Classification," arXiv.org, 22-Nov-2017. [Online]. Available: https://arxiv.org/abs/1711.08200.
[12] C. Li, Q. Zhong, D. Xie, and S. Pu, "Skeleton-based action recognition with Convolutional Neural Networks," arXiv.org, 25-Apr-2017. [Online].
Available: https://arxiv.org/abs/1704.07595.
[13] X. Gao, W. Hu, J. Tang, J. Liu, and Z. Guo, "Optimized skeleton-based action recognition via sparsified graph regression," arXiv.org, 15-Apr-2019. [Online]. Available: https://arxiv.org/abs/1811.12013.
[14] R. Vemulapalli and R. Chellappa, "Rolling Rotations for Recognizing Human Actions from 3D Skeletal Data," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 4471-4479, doi: 10.1109/CVPR.2016.484.
[15] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal Speed and accuracy of object detection," arXiv.org, 23-Apr-2020. [Online]. Available: https://arxiv.org/abs/2004.10934v1.
[16] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, "Microsoft Coco: Common Objects in Context," arXiv.org, 21-Feb-2015. [Online]. Available: https://arxiv.org/abs/1405.0312.