Vehicle Re-Identification and Tracking using Deep Neural Network

Shubham Jiwtode

Abstract: Vehicle re identification is one of the challenging computer vision job, the minute variation in car of same color and variation in illumination, view angle, background makes it difficult to recognize same car in different cameras. This paper provides a simple but not trivial solution to address this problem, model used in this paper consist of a pre-trained model that can detect cars, a Siamese model to detect the car of our interest in that video frame and median flow tracker method from OpenCV that tracks object (car in our case) with low computation power. This model provides an efficient way to perform the task such that GPU power is used when needed by the system. Model was successfully able to identify the vehicle and draw some useful information like speed and direction of the vehicle. Reidentifications of vehicle can be used in many areas from video surveillance, assisting drone to follow a vehicle, vehicle pursuit. The work done in this paper was limited due to computation power and good dataset. We will also cover what can be done further to improve the model efficiency and raise the accuracy.

***Index Terms*— Vehicle Re-identification, Convolutional Neural Network, Siamese Network, YOLO**

I. INTRODUCTION

Automotive industry has been exploiting computer vision for various task from vehicle detection, vehicle speed, vehicle count, autonomous vehicles. Vehicle re identification is also one of the most valuable and challenging tasks, the work done under this paper was inspired from face reidentification but unlike face reidentification, vehicle reidentification is quite under studied. Vehicle reidentification is mostly useful for the government or police departments for surveillance or to ease the vehicle pursuit. When working on images or videos the first type of neural network that comes to everyone's mind is Convolutional Neural Network. CNN are used widely due to its feature extraction, it effectively considers the information embedded in adjacent pixel to down sample it by convolution and then uses prediction at the end. Deep learning being the method used in this paper, infamous for its huge data requirements. The whole work was limited due to scarcity of database for this particular job, acute pair database forced us to limit the training of Siamese network which is responsible to detect the query vehicle in the given frame. Performing labelling to build a pairwise dataset for cars consumes lots of time and human resource. The dataset used in this paper was the closest dataset we were able to gain. Second limitation was

computation power unable to run convolutional neural network on my system I was forced to use google colaboratory, but it has its own limitation when it comes to cloud storage that is provided to us to run the required code. Thus, to get around this issue we used a pre-trained network so that we don't have to train a system just for vehicle detection, as there are already many models available that produce satisfactory results and

Shubham Jiwtode, is now in Electrical Engineering and Computer Science Department, Syracuse University, Syracuse, NY 13210 USA (e-mail: srjiwtod@syr.edu).

makes sense to use it. Finally, we have used object tracker by OpenCV, the purpose of using this algorithm was to lower the computation usage and make the whole system efficient and as reliable as possible. After the system identifies vehicle, we can extract some more information from the special domain using the coordinates in the bounding box. The bounding box are 2d and defined in (x, y, w, h) format. System successfully use these coordinates to show the direction of the identified vehicle and the speed with which it is travelling. Now let's dive deep into the timeline of this project from literature survey to actual implementation, results obtained from the system and also will try to point out future scope of this project and some future work that would improve system overall.

II. Related Work

In this section we will look deep into the technologies used in our system implementation.

A. Convolutional Neural Network

CNN are neural network from deep learning class most widely used on visual related task. These network tries to replicate human visual cortex. CNN consist of several hidden layers, hidden layer composes of convolution layers, pooling layers, fully connected layer. All the values are aggregated into a loss function and back propagation update weights in a loop. Input to Convnet is a 3D matrix height, weight and depth. While output is a 1*1*x where, x is the number of classes to detect. Once an input is fed to the nets 2d kernel or filter is applied on each channel of the image. These filters are nothing but one type of feature with attribute size, stride, zero-padding. Activation layer play a n important role in neural network, these activation functions are nonlinear, and they are the one which decides when a particular node is fired. These takes weighted sum with bias as input. Pooling layer down samples the input matrix into a small matrix, basically it takes few adjacent values and give a value depending on type of pooling max pooling or average pooling. Lastly there are fully connected layer where in each node is connected to every other node in next layer. These layer does the classification job. Output of this layer gives the probability of a input belonging to a class. Convnet don't need any preprocessing. Thus, lowers human efforts as compared to traditional algorithm. Summarization is as described below. [11]

- Accepts a volume of size $W1\times H1\times D1$
- Requires four hyperparameters:
  1. Number of filters K,
  2. their spatial extent F,
  3. the stride S,
  4. the amount of zero padding P.
- Produces a volume of size $W2\times H2\times D2$W2×H2×D2 where:
  1. $W2=(W1-F+2P)/S+1$
  2. $H2=(H1-F+2P)/S+1$ i.e. width and height are computed equally by symmetry)
  3. $D2=K$D2=K
- With parameter sharing, it introduces $F\cdot F\cdot D1$ weights per filter, for a total of $(F\cdot F\cdot D1)\cdot K$ weights and K biases.
- In the output volume, the d-th depth slice (of size $W2\times H2$) is the result of performing a valid convolution of the d-th filter over the input volume with a stride of SS, and then offset by d-th bias [11].

B. Yolo

Yolo is a single unified real time object detection architecture. Architecture consist of single convnet model output of this net is bounding boxes and associated classes with probabilities. Network uses features from entire image to predict each bounding box. YOLO enables high speed detection while maintaining high average precision. Yolo divides the input image in S*S grid. If the center of object falls into grid cell that grid is responsible for detection of that object [1]. Each cell predicts bounding boxes and confidence for those boxes.

These scores represent the confidence of model on occurrence of that particular object we define confidence score as probability of object*intersection over union. Each predicted bounding box consist of parameters (x, y, w, h) coordinates and confidence. Where x and y represent center of box relative to grid cell, w and h are the width and height of frame respectively. Below is the model architecture of yolo.

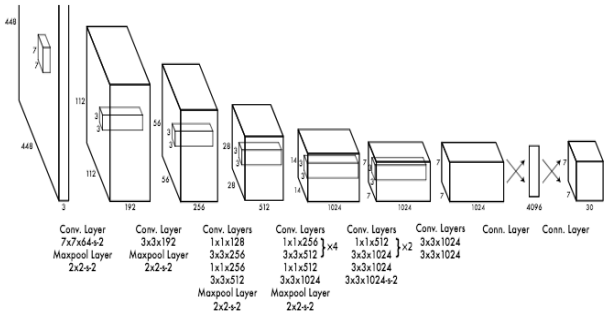


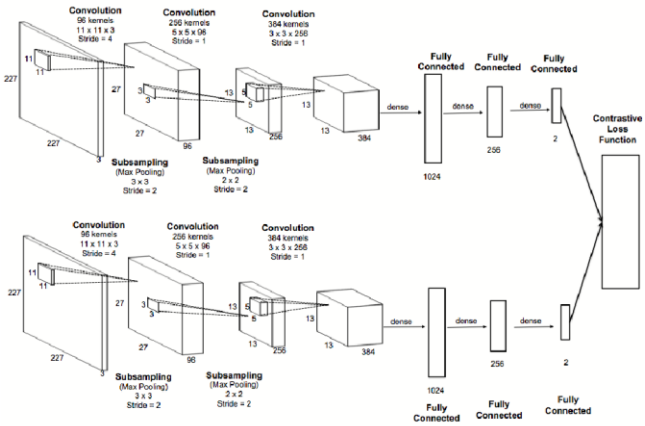
Fig. 1: Yolo V1 Neural Net architecture [1]

Network has 24 convolutional layers and fully connected layer, input to this network is 448*448 color image and the output is 7*7*30 vector. For this model we have used predefined weights from darknet website.

Yolo has some limitation like it can only have two bounding boxes per grid cell and only one class. Yolo is trained on a loss function corresponding directly to performance and entire system is trained jointly. [14]

C. Siamese Network

Learning features and finding the similarity between two frames or finding existence of an object in a frame can be expensive computationally. Apart from computation expenses these networks also require huge amount of data for training. In such a case where you know the input data for the network is limited to few images Siamese network does a good job of finding the similarity between the query image and the current frame. There is a unique implementation of Siamese network called one shot learning. One shot learning says training of a model can be done even if you just have one image of each class. We use this network in our implementation to detection if our query vehicle is in the given frame. The architecture of Siamese network is simple the network comprises of two identical neural networks each taking two images, the end output of both the images are fed to a custom contrastive loss function which gives out the similarity between two images that were fed to the sister neural network.



F ig. 2: Architecture of Siamese Network [12]

The contrastive loss function is used because we intent to differentiate two images, whereas common classification function won't prove that effective in this case.

D. Median Flow Tracker

Imported from OpenCV object tracking API. This tracker tracks an object at a rate well above 60 fps if satisfying bounding box ids provided. The reason and advantage of using this is that it requires very low computation power. The disadvantage of this tracker proved advantageous for our model, this model doesn't try to find the object once lost and this is perfect for us to kick in over neural network to get back the bounding box coordinates and detect the query object which ensures we are tracking correct object.

E. Object Re identification

Object reidentification refers to tracking object in different scene as they appear in different cameras. There are many applications for this like video surveillance where going through all the video footage can be a cumbersome task, deploying object re identification model on those video footages will not only save human efforts but also save lots of time as systems can give going without taking pause. This can also be used in Realtime system like vehicle pursuit where real time cctv footage will be scrutinizing to locate the vehicle and predict the possible future location by deriving speed and direction of vehicle. The fact that it has wide range of application doesn't spare it from challenges. The variation in angle, illumination, orientation makes similar object look a bit different. Thus, while design the model we need to consider these variations such that the model doesn't give up its accuracy. Re identification has received major attention mainly due to advance in algorithm and deep learning filed. In this paper we will be using two types of deep neural net to achieve our goal of object detection. The model proposed in this paper will try to answer this problem with limited dataset and computation power but still being satisfactorily accurate. Paper also showcases potential of model and room for improvement in future with large dataset and equally good computation power.

III. PROPOSED APPROACH

With all the above described technologies we try to build a system that will use power efficiently. For whole project we have used jupyter notebook with keras 1.2.2 and TensorFlow as backend. Frames from video file were extracted using moviepy. Before feeding these frames to our model they were resized to 448*448. The ration was chosen such that not much information is loss while downsizing the image.

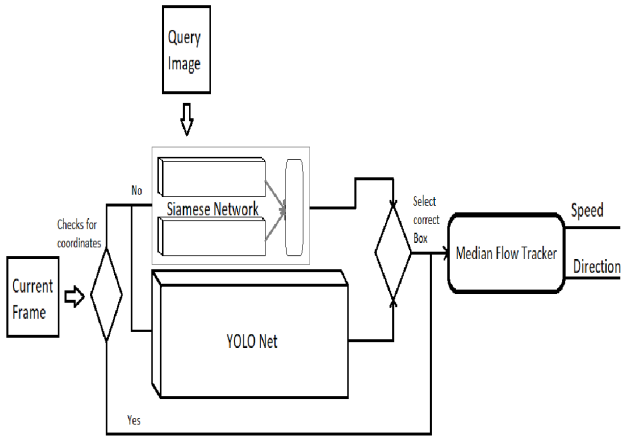


Fig. 3: Proposed Model architecture

As shown in the proposed architecture the downsized image will be passed to Siamese network and also through a pre-trained model for car detection i.e. YOLO in our case, model has been loaded with the weights from darknet website. The pretrained model passes this image from multiple layers of convolution, max pooling, then finally fully connected layer. The output of this layer was processed to draw out the boxes around the cars. While yolo detects cars in frame our Siamese network working simultaneously compares the query image with the frame to find out the similarities between them. Once it has detected the presence of our interest car. We can get the possible coordinates of our car of interest. Thus, output of these two networks is a bounding box. This process is iterated over all the frame until and unless we get a bounding box. Once we get a bounding box, we pass these coordinates to object tracker algorithm (median flow tracker) present in OpenCV library. Once the tracker receives the coordinates it bypasses the neural network for all preceding iteration. If unfortunately, the tracker loses the car, the neural nets are triggered in next frame iteration. This process was performed throughout the video. Hence in this way we succeeded in re-identifying the car. After detecting the car we intend to draw some useful information from the identified car. For this we have located a region of interest in our frame. ROI is located at the height of 450-

pixel and 600-pixel height. Once the vehicle passes through the ROI if note the frame numbers of entry and exit of ROI. These frame number and current FPS gives us the time car took to travel from point of entry to point of exit. If calibrated properly we can get the physical distance between the entry and exit point. Hence with simple distance by time formula we can get the speed of vehicle. Once the speed is detected, we can use same bounding box coordinates to understand the direction in which the car is moving.

- (L, R, T, B) and x coordinates bounding box and frame number at the time of entry in ROI respectively.
- (L', R', T', B') and x' coordinates bounding box and frame number at the time of entry in ROI respectively.
- Time to cross ROI= **(||X - X'||) / frame rate**
- Distance travelled(vertically) = **(||L-L'||)/ ROI time**
- If T > T', car travelling downwards
- If T'> T, car travelling upwards

These all factor can help us not only to identify the car but also predict its future location. Due to this many of our resources can be saved and utilized in some other task if needed.

IV. Results

The system proved to be successful in identifying the car from one query image. Using Yolo, we were able to achieve a frame rate of 20 FPS while median flow tracker was running on 50+ FPS. Along with vehicle identification we were able to track the vehicle and draw out its speed and direction in which it is moving. Below are results of vehicle detection without deployment of Siamese network and reidentification with yolo and Siamese network.

Fig. 4a: car detection output screenshot. (**Left**) 4b: car re identification output screenshot. (**Right**)

V. Future Work

Work till now was done with acute dataset and limited computation power. In this section we provide some suggestion that will improve the performance and accuracy of the system. First and foremost, we can replace the pretrained Yolo model with our own model which can detect cars. Yolo is trained and design to detect over 20 classes while we are only interested in cars. This will save lost of computation power and will also improve the accuracy and performance. Similarly, we can train the Siamese network effectively with pair dataset while keep in consideration some distinguishable car features. A well-trained Siamese network with contrastive loss will be able to identify the car more quickly as compared to current case. One more work we can do is run model on multiple GPU/CPU. More power system will run the reidentification model (YOLO and Siamese) and the less powerful system will run the tracker. Thus, this will make the whole system very efficient.

References

[1] Redmon, Joseph, et al. "You only look once: Unified, real-time object detection." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.

[2] Jang, Daniel Marcus, and Matthew Turk. "Car-Rec: A real time car recognition system." *applications of computer vision (WACV), 2011 IEEE Workshop on*. IEEE, 2011.

[3] AbdelMaseeh, Meena, et al. "Car make and model recognition combining global and local cues." *Pattern Recognition (ICPR), 2012 21st International Conference on*. IEEE, 2012.

[4] Lagunas, Manuel, and Elena Garces. "Transfer Learning for Illustration Classification." *arXiv preprint arXiv:1806.02682*(2018).

[5] Yosinski, Jason, et al. "How transferable are features in deep neural networks?." *Advances in neural information processing systems*. 2014.

[6] Sochor, Jakub, Jakub Špaňhel, and Adam Herout. "Boxcars:

[7] Improving fine-grained recognition of vehicles using 3-d bounding boxes in traffic surveillance." *IEEE Transactions on Intelligent Transportation Systems* (2018).

[8] Coifman, Benjamin. "A new algorithm for vehicle reidentification and travel time measurement on freeways." *Proc. of the Fifth Conference on Applications of Advanced Technologies in Transportation*. 1998.

[9] Kanacı, Aytaç, Xiatian Zhu, and Shaogang Gong. "Vehicle reidentification by fine-grained cross-level deep learning." *BMVC AMMDS Workshop*. Vol. 2. 2017.

[10] Zapletal, Dominik, and Adam Herout. "Vehicle re-identification for automatic video traffic surveillance." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2016.

[11] Garcia, Christophe, and Manolis Delakis. "Convolutional face finder: A neural architecture for fast and robust face detection." *IEEE Transactions on pattern analysis and machine intelligence* 26.11 (2004): 1408-1423.

[12] Rao, Sanjeev Jagannatha, Yufei Wang, and Garrison W. Cottrell. "A Deep Siamese Neural Network Learns the Human-Perceived Similarity Structure of Facial Expressions Without Explicit Categories."

[13] https://www.docs.opencv.org/3.1.0/d2/d0a/tutorial_introduction_to_tracker.html

[14] https://github.com/udacity/CarND-LaneLines-P1

[15] https://pjreddie.com/darknet/yolo/