Pedestrian Detection in Real Time by Using Deep Learning Algorithms

Abstract

Pedestrian detection in real-time can be used in many situations as it has the ability to determine the type of objects, counts of objects and motion of objects. This technique can be used for surveillance to detect unwanted people via surveillance video; it can be used in advanced robotics for detecting objects around it, similarly it is used in assistance in driving a car and it can be used for self-driving car technologies.

In this paper, I have explored various Deep Learning algorithms specifically based on convolution neural networks and compared their results. I have used YOLOv2 algorithm for object detection with various feature extraction CNN models such as DarkNet19, SqueezeNet and MobileNet which are available for the backend.

Introduction

A statistic published by WHO states that every year around 1.5 million people die because of road accidents. Counts for non-fatal injury is higher at 20-50 million people which results in financial problems in the form of hospital treatments, family members missing work that causes an estimated cost of 3% GDP for the average country ("Road Traffic Injuries," n.d.). Therefore, it has become important to find a well-oiled solution for pedestrian detection. There have been several solutions proposed in last decade for automatic vehicles.

Pedestrian detection is challenging compared to object detection because of the context, light, posture, obstacle, crowd density in a frame. It is not only the accuracy, but the speed of an algorithm is critical for choosing an algorithm to detect pedestrian in real time. Deep Learning has recently become popular with the excess availability of data and the recent emergence of high-performance parallel computing systems. Presently, we can easily identify the content of an image easily or use natural language processing with the help of Deep Learning methods. In Deep Learning, a structure is composed of multi-layered artificial neurons connected to each other through a mathematical function where each neuron takes a linear combination of feature parameters and gives an output based on an activation function.

One of the approaches of Deep Learning is Convolution Neural Networks (CNN) or ConvNet which takes an image which is converted into a vector that is acceptable to an algorithm as an input and assigns parameters (biases and weights) depending on features of an object in an image to differentiate it from other objects in the same image. It is inspired by the human brain and visual context where an individual neuron acts only for a stimulus; it is limited to a visual field which is also known as the Receptive Field. CNN is used for image classification, enhancement, object detection, tracking and many more related fields. Object detection using Deep Learning is performed in two steps, first it will classify the objects in the image based on feature extraction algorithms, it will then locate objects in the image using regression approach. In this paper, I have used Deep Learning algorithms to detect pedestrians. Object detection has a vast number of use cases in the field of computer vision. I have analyzed accuracy, parameter counts and speed of several Deep Learning models such as the YOLOv2 algorithm and compared their results. Developing a Deep Learning model is a time and resource computing task, so I have utilized Transfer Learning and use trained weights to analyze Caltech Pedestrian Dataset. I have used Keras with Tensorflow 2 as the backend to train and detect objects in the dataset.

Literature Review

Over the last few decades many solutions have been proposed for pedestrian detection; researches have used methods such as hand-coded features and used classifiers such as SVM to detect a face in real time, including Viola-Jones algorithm (Viola & Jones, 2001), Histogram of Oriented Gradients (HOG) (Dalal & Triggs, 2005). Viola-Jones algorithm finds a hand-coded area on the face like eye, nose, mouth and their relationships among them, the face related vectors feed into the SVM algorithm to detect whether an image has a face or not. On the other hand, HOG uses pixel level details. For each pixel, it sees surrounding pixels and lays down an arrow in the direction of darker area. This arrow is called the gradient. This process is repeated for each pixel. The image is broken into a 16x16 smaller squares and within each square it sees the large numbers of arrows pointing to a particular direction and replaces that square with that particular arrow. After repeating the same process for each smaller square, the result image captures the basic information of the original image.

These solutions have one major drawback which is the manual extraction of features which causes low accuracy and slow execution. On the other hand, extraction of features automatically in deep learning ensures high accuracy and speedy solutions than could be used in real time systems. A new approach has been proposed by using R-CNN (Girshick, Donahue, Darrell, & Malik, 2014). It extracts ~2K regions by using selective search (Uijlings, van de Sande, Gevers, & Smeulders, 2013). All extracted features are then further classified using SVM; this algorithm also predicts the offset of an object within the box.

However, it is slow, it takes ~ 47 seconds so it cannot be used in real time systems. Moreover, selective search is a static approach, so no learning happens at runtime. Another attempt was made in Fast-RNN (Girshick, 2015); instead of feeding region of proposal, an image and a set of objects proposals feed into CNN. After extracting feature maps, these maps feed into ROI pooling layer and after that a SoftMax layer predicts proposed region's class and four offset coordinates for bounding box. It has better performance with respect to Fast-CNN but still it is slower because of the region selection. Faster-RNN (Ren, He, Girshick, & Sun, 2016) took another approach, instead of using selection search for finding proposal region, it allows the network to learn the region for proposal. An image is fed into the CNN network and regions proposals are predicted by neural network added separately. Then those regions are used in the ROI layer which is then further used to predict a class for the proposed region and four offset coordinates of bounding boxes. It has an improved speed over R-CNN and Fast-CNN is a good candidate to detect objects in real time.

There is another method for the object detection which is Single Shot MultiBox Detector: SSD (Liu et al., 2016). It processes once to detect many objects within the subjected image while other solutions take two steps, one for producing region proposals and another for detection of object for each proposal. SSD300 had shown a result, 59 FPS with mAP 74.3% on VOC2007. A different approach is taken with YOLO (Redmon, Divvala, Girshick, & Farhadi, 2016): "you only look once". Instead of breaking an image into region proposals and finding objects in those regions which has high probabilities for detecting objects, this network takes a single image and predicts both bounding boxes and class probabilities for each of these boxes. It has ability to detect video with 45 frames per second, but it has a limitation to detect small or densely placed objects. A successor of YOLO is YOLOv2 or YOLO9000 (Redmon & Farhadi, 2016). It is the current state-of-the-art method and has a many fold improvement over YOLO and can detect more than 9000 objects in a single image. It has 76.8% mAP with 67 FFS and 78.6% mAP with 40 FPS. Next successor is YOLOv3 (Redmon & Farhadi, 2018). It uses DarkNet-53 with a better object detector. YOLOv4 (Bochkovskiy, Wang, & Liao, 2020) is a latest version. It has achieved 43.5% mAP with speed of 65 FPS on Tesla V100 for COCO dataset.

Methodology

Pedestrian detection requires identifying objects in an image, it should return pedestrians with its location in an image. So, a solution is to perform classification before detecting an object in an image in question. CNN has been used widely to classify objects in the image, it gives the relative probability of objects presents in the image by using the SoftMax function. CNN uses feature extraction using layers of neural networks and weights to minimized loss.

To detect pedestrians inside the image, an object needs to be localized i.e. its position relative to some reference point, generally in a rectangular box. Finding a location is a type of regression problem where a box predicts an object based on confidence, extent of the object presents inside the box, intersection over union (IOU). It gives coordinates (x, y) of box's center with height, confidence score and class of the object with probability.

The YOLO is an algorithm to calculate the position of an object within the image however it has precision issues. The YOLO architecture is designed to recognize objects in real-time. However, it has a limitation (49, one for each bounding box of a grid of size 7x7) to detects no of objects in the image. Another problem is that it can only detect one object per cell; therefore, if there are two objects close to each other than the algorithm would only be able to recognize one of them, similarly it can detect similar objects many times over.

YOLOv2 increases accuracy with speed for detecting and classifying objects in the image in real time. These are the key areas where improvement had happened:

1) It uses batch normalization to improve accuracy of 2% mAP and it also helps in regularization of model without using dropout.
2) It uses a single convolution layer to detect and predict bounding box in a single attempt using predefined anchor boxes instead of full connected two layers of YOLO.
3) It uses the anchor boxes which are obtained using k-means clustering on given bounding box dataset and produces five bounding boxes.
4) It calculates 5 values (t_x, t_y, t_w, t_h, t_o) and uses sigmoid activation function to calculate coordinates for every bounding box (bx, b_y, b_w, b_h) where (cx, cy) is the top left corner of the anchor box and (p_w, p_h) is the width and height of the anchor box.

   $b_x = \sigma(t_x)$

   $b_y = \sigma(t_y)$

   $b_w = p_w e^{tw}$

   $b_h = p_h e^{th}$

and box confidence can be calculated with following formula where P_r is probability of an object for the box and IoU is intersection over union for ground truth box and predicted box.

$P_r(\text{object}) \times \text{IoU}(b, \text{object}) = \sigma(t_o)$

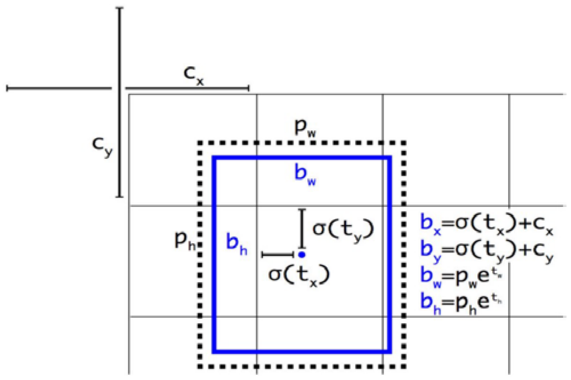


Fig. 1 One of the anchor boxes shown in dotted line, blue is a predicated box.

YOLOv2 algorithm uses DarkNet19 as the feature extractor but for my experiment I have changed it to SqueezeNet (Iandola et al., 2016) and MobileNet (Howard et al., 2017) for feature extraction and measured and compared their performances.

SqueezeNet:

It is a small CNN and achieves accuracy similar to AlexNet on ImageNet dataset and reduced parameters uses by 50 times. It can be used on devices with constrained resources like memory and CPU.

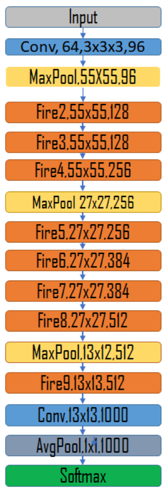
Followings are the main features implemented to achieve SqueezeNet:

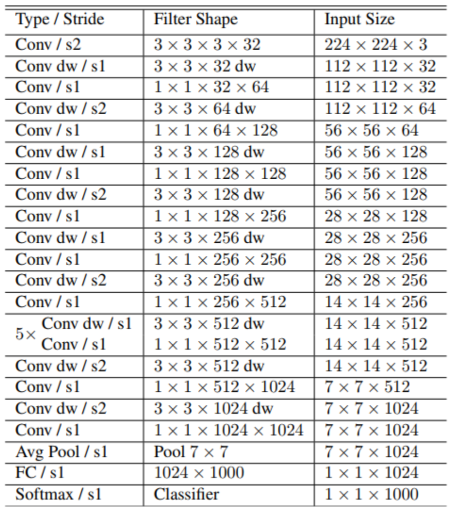
1) It uses 1x1 filters instead of 3x3 filters or reduces the count of input channels.
2) To maintain the accuracy of it, it downsamples late in network.

Fire Module is the main constitute of SqueezeNet, it is a combination of squeeze convolution layer (only 1x1 filters) and then feeds that into expand convolution layer (mix of both 1x1 and 3x3 filters).

MobileNet:

It is also efficient in terms of speed and size and suitable for embedded devices. It uses depthwise convolution which is a single filter for each color channel rather than combining all three and flattening it, this has the effect of filtering the input channels.





Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
5× Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
5× Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

Fig 2. SqueezeNet model Fig. 3 MobileNet model

Experiments & Results

For the implementation of the model and an algorithm, I have used Keras and TensorFlow2. I have used Conda for my environment and dependencies management.
I have used Caltech Pedestrian Dataset; it has 250,000 frames with total 350,000 bounding boxes approximately video of 137 minutes with 2300 unique pedestrian. Since training a model is a time and resource consuming task therefore, I have used Transfer Learning. I have used weights pretrained on the Raccoon Dataset (Tran, 2017/2020) and transferred learning to Caltech Pedestrian Dataset. I have trained the model and predicated bounding box using SqueezeNet and MobileNet as features extractor and here are the results:

Table 1. Comparison of different backend used with YOLOv2.

Algorithm	**mAP (%)**
YOLOv2 + DarkNet19	0.6677
YOLOv2 + SqueezeNet	0.7432
YOLOv2 + MobileNet	0.5534

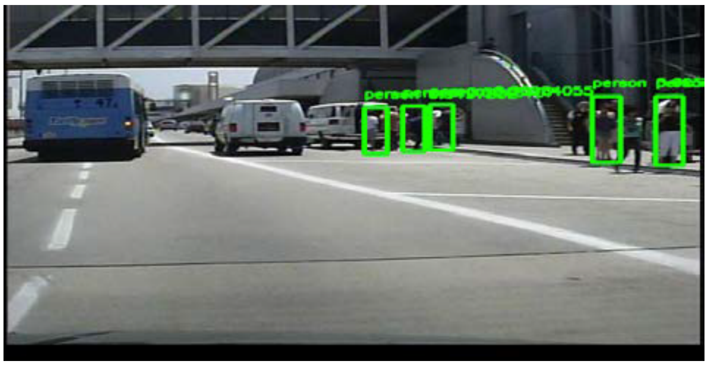


Fig. 4 Real time detection of vehicles.

Conclusion

I have used YOLOv2 for real time detection of pedestrians using Caltech Pedestrian Dataset and shared results. It is safe to use YOLOv2 to detect pedestrian in real time for devices with limited resources like mobile and hybrid devices with MobileNet and SqueezeNet because of their low memory footprints of networks and acceptable accuracy they produce.

Timetable for Completion

Task	**Deadline**
Final decision on the topic, create research questions	February 24th, 2020
Literature review	March 30th, 2020
First round of testing	April 10th, 2020
Analyse and collate the test results	April 20th, 2020
Second round of testing	May 10th, 2020
Analyse, collate and compare new results to previous ones	May 27th, 2020
Present final results and report	June 3rd, 2020

REFERENCES

Bochkovskiy, A., Wang, C.-Y., & Liao, H.-Y. M. (2020). YOLOv4: Optimal Speed and Accuracy of Object Detection. *ArXiv:2004.10934 [Cs, Eess]*. Retrieved from http://arxiv.org/abs/2004.10934

Dalal, N., & Triggs, B. (2005). Histograms of oriented gradients for human detection. *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, *1*, 886–893 vol. 1. https://doi.org/10.1109/CVPR.2005.177

Geiger, A., Lenz, P., & Urtasun, R. (2012). Are we ready for autonomous driving? The KITTI vision benchmark suite. *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 3354–3361. https://doi.org/10.1109/CVPR.2012.6248074

Girshick, R. (2015). Fast R-CNN. *ArXiv:1504.08083 [Cs]*. Retrieved from http://arxiv.org/abs/1504.08083

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. *ArXiv:1311.2524 [Cs]*. Retrieved from http://arxiv.org/abs/1311.2524

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., … Adam, H. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *ArXiv:1704.04861 [Cs]*. Retrieved from http://arxiv.org/abs/1704.04861

Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., & Keutzer, K. (2016). SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size. *ArXiv:1602.07360 [Cs]*. Retrieved from http://arxiv.org/abs/1602.07360

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). SSD: Single Shot MultiBox Detector. *ArXiv:1512.02325 [Cs]*, *9905*, 21–37. https://doi.org/10.1007/978-3-319-46448-0_2

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. *ArXiv:1506.02640 [Cs]*. Retrieved from http://arxiv.org/abs/1506.02640

Redmon, J., & Farhadi, A. (2016). YOLO9000: Better, Faster, Stronger. *ArXiv:1612.08242 [Cs]*. Retrieved from http://arxiv.org/abs/1612.08242

Redmon, J., & Farhadi, A. (2018). YOLOv3: An Incremental Improvement. *ArXiv:1804.02767 [Cs]*. Retrieved from http://arxiv.org/abs/1804.02767

Ren, S., He, K., Girshick, R., & Sun, J. (2016). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *ArXiv:1506.01497 [Cs]*. Retrieved from http://arxiv.org/abs/1506.01497

Road traffic injuries. (n.d.). Retrieved May 21, 2020, from https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries

Tran, D. (2020). *Datitran/raccoon_dataset* [Jupyter Notebook]. Retrieved from https://github.com/datitran/raccoon_dataset (Original work published 2017)

Uijlings, J. R. R., van de Sande, K. E. A., Gevers, T., & Smeulders, A. W. M. (2013). Selective Search for Object Recognition. *International Journal of Computer Vision*, *104*(2), 154–171. https://doi.org/10.1007/s11263-013-0620-5

Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, *1*, I-511-I–518. https://doi.org/10.1109/CVPR.2001.990517

Responses to Reviewers Comments on

Project Proposal

COMP825-2020 Pedestrian Detection

Reviewer 1

Question 1.1: The abstract is slightly longer, and please reduce it to around 100 words. Peer review is too monotonous. It merely lists technical terms without analysing the work of other literature.

Answer 1.1: This has been updated.

Question 1.2: To consider video to detect objects

Answer 1.2: I have implemented object detection for a video using Keras implementation

Reviewer 2

Question 2.1: There is no related work or literature review section it is all included in introduction.

Answer 2.1: Introduction and Literature reviews have been updated.

Question 2.2: There is no timetable for completion or Expected research output and ways of quality assurance mentioned

Answer 2.2: Those sections have been updated.

Reviewer 3

Question 3.1: There is no separated "literature review/related work" section.

Answer 3.1: This has been updated.

Question 3.2: The bibliography is not in APA format.

Answer 3.2: This section has been updated.