Pedestrian Recognition Using Neural Networks

Dax Patel, Parth Patel

Semester: Fall 2020

Task V Draft

Introduction

The focus of this paper is on improving the safety of autonomous vehicles by using machine learning to identify pedestrians. The development of autonomous vehicles has increased greatly over the past few years. As these vehicles become more readily available it is vital that these vehicles are safe for the people that will be using them. In order to be safe for both pedestrians and drivers, these vehicles must be able to quickly and accurately identify pedestrians.

The most efficient way for vehicles to identify pedestrians is by having them learn the normal shapes of humans and then train it so that it will be able to identify pedestrians in various different conditions. In a recent article, various different methods to identify pedestrians like YOLO(you only look once) and R-CNN(region convolutional neural network) are described and compared [7]. Most of these methods have one thing in common, they all evolve from convolutional neural networks.

These previous methods all have their improvements and drawbacks compared to convolutional neural networks. Some are faster and made to be used in real-time while others are more accurate but take longer to process images. The purpose of this paper is to start from the beginning and make a convolution neural network so that it can be used as a back-bone for these other object detection techniques.

Related Work

Over the years, there have been several different kinds of techniques that have been developed to address obstacle detection from an image. In the paper *"Image Segmentation Techniques",* the authors compare the evolutions of object detection in images [7]. They start with Image Segmentation. In this technique, the image is divided into segments by identifying pixels in regions that have a rapid transition in intensity. Edge detection can be carried out in two possible ways: Gray Histogram Technique and Gradient-Based Method[7]. In the histogram-based method, a histogram is created using some threshold value of pixel intensity. Then local maximums and minimums are used to determine edges. This technique is difficult as the histogram could turn out to be uneven. In the latter approach, gradients for an image are used. Gradients are the first derivative of an image that can be represented as a function. Edges are defined as spaces where an abrupt change is detected by identifying the change in gradient values. This can be done by convolving gradient operators with the image and finding places of rapid transition between gradient values.

The second approach the author talks about is Segmentation based on the Artificial Neural Network. According to the explanation, each neuron is used to represent a pixel, which converts the problem into smaller sections for the neural network to handle [7]. The neural network has two main steps: feature extraction, and image segmentation based on a neural network. The significance of feature extraction is that it helps reduce the size of the input that is fed into the network. Each neuron gets input from the neuron before it. The data received depends on weight. When training, the weight is assigned a random number. Then the network output is tested against the labeled dataset. If the output is not correct, the weights are adjusted by a small out, and the output is tested again. This process continues until the accuracy reaches an expected level.

Over the years, efforts have been made to enhance the basic neural networks to make them fast and efficient. An example of this is the research conducted by Tianrui Liu and Tania Stathaki [10]. They propose a Faster Region-Based Convolutional Neural Network - R-CNN. Research done by authors concludes that in a simple R-CNN, they use a selective search approach to extract "region" from the image as a feature extraction step. Then this "extracted region" is fed into the network for training and

prediction [10]. There are two problems with this approach; the first happens to be the speed, and the second one is identifying trees and electric poles as a person which is a false negative and not good for autonomous vehicles [10]. Authors conclude by making suggestions on a CNN which has a parallel layer of Region Proposal Network (RPN). This parallel layer gets features from objects that can be used by CNN when identifying objects in later images. This allows CNN to compute feature maps while also getting features of objects from the RPN at the same time.

Another type of CNN is "Mask R-CNN". The study done by Kaiming He, et .al [3] considers some popular methods for object recognition from an image like Histogram Oriented Gradient, Image Segmentation, and R-CNN. In this approach, there is an additional branch which calculates the segments on each region of interest. What this does is shade the target object in the output of the image along with a drawing box around the target object. The shading part is called the "pixel level mask" of an object [3].

Shih-Chieh Lin, et.al [9] describes a more complex version of CNN called YOLO(You Only Look Once). YOLO takes an image and separates it into a grid and each part of the grid must be a part of at least 5 bounding boxes. These bounding boxes are combined and eliminated depending on how they interact with each other. Bounding boxes are combined whenever a large number of boxes overlap. Bounding boxes are eliminated if few overlaps occur. The image is only looked at once hence the name and as a result, it can identify objects very quickly. YOLO accomplishes this speed by speeding up how neural networks function by reducing the number of convolution layers by three, reduce the number of filters to reduce dimensions of feature maps, and add a residual structure to use only a fraction of the feature map to get an accurate reading of an image [9].

Dataset

There are many different datasets available for person recognition like cityscapes data use. The dataset used for this project is Penn Fudan Pedestrian and it contains over 150 that are labeled with pedestrians[12]. These images are then split for training and testing. About 90% of the images were used for training and the rest are used for testing. This dataset is labeled in the Pascal format. A python script was used to extract the coordinates of the bounding boxes and convert them to a CSV file. Each row of the CSV file contains the name of the image, dimensions of the image, class of the object presented, and coordinates of the bounding box.

A	B	C	D	E	F	G	H
filename	width	height	class	xmin	ymin	xmax	ymax
FudanPed00001.png	559	536	person	158	175	323	441
FudanPed00001.png	559	536	person	408	165	546	496
FudanPed00002.png	455	414	person	67	82	191	383
FudanPed00002.png	455	414	person	153	103	221	370
FudanPed00003.png	479	445	person	295	133	444	419
FudanPed00004.png	396	397	person	166	57	324	342
FudanPed00004.png	396	397	person	8	58	56	179
FudanPed00005.png	335	344	person	190	51	321	340
FudanPed00005.png	335	344	person	2	48	46	162
FudanPed00006.png	385	426	person	204	104	357	390

Figure 1 Labeled Converted to CSV

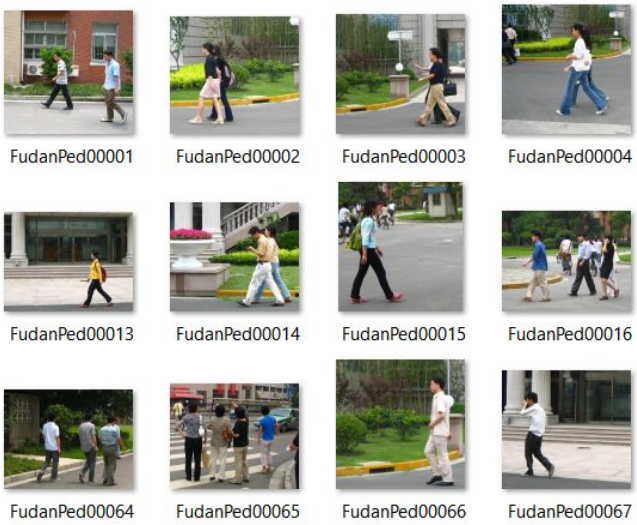

Figure 2 Images from Dataset

Algorithm

Figure 3 shows each layer of the convolution neural network. The input and output of each layer can be identified.

```
Model: "sequential_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_17 (Conv2D)           (None, 224, 224, 64)      1792
_________________________________________________________________
conv2d_18 (Conv2D)           (None, 224, 224, 64)      36928
_________________________________________________________________
max_pooling2d_17 (MaxPooling (None, 112, 112, 64)      0
_________________________________________________________________
conv2d_19 (Conv2D)           (None, 112, 112, 128)     73856
_________________________________________________________________
conv2d_20 (Conv2D)           (None, 112, 112, 128)     147584
_________________________________________________________________
max_pooling2d_18 (MaxPooling (None, 56, 56, 128)       0
_________________________________________________________________
flatten_5 (Flatten)          (None, 401408)            0
_________________________________________________________________
dense_21 (Dense)             (None, 128)               51380352
_________________________________________________________________
dropout_15 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_22 (Dense)             (None, 64)                8256
_________________________________________________________________
dropout_16 (Dropout)         (None, 64)                0
_________________________________________________________________
dense_23 (Dense)             (None, 32)                2080
_________________________________________________________________
dense_24 (Dense)             (None, 4)                 132
=================================================================
Total params: 51,650,980
Trainable params: 51,650,980
Non-trainable params: 0
```

Figure 3 Model Summary

Before the program can identify pedestrians it must first be trained with the training dataset. Each image in the dataset will be separated into RGB(red green blue) values. After each pixel is assigned a value then the program will try to find any large gaps in color values to find edges for different shapes in the image. In general, a large change in color values means that something in the image is a completely different color and darkness than the surrounding pixels. This means that in most cases there are separate objects in the image. After a feature map is made showing the general shape of the image the program will pool the image and splits it into smaller sections. This step should act similarly to how a person would look at an image by separating the image into sections and looking at the values in each section and

finding a value that corresponds to that section. After obtaining the smaller sections the values shall be flattened before going into the second layer of CNN. Then the data will go through a hidden layer that contains several nodes. The number of nodes affects speed and accuracy and would be adjusted during model development. These hidden nodes will create a weight model that associates pixels with pedestrians. These weights are saved and can be used later on for any image passed to the program. The goal of this model is to provide robustness and faster training time for a larger system that would use this as a backbone.

Implementation

Before the images can be passed into the CNN, the program first loads in the training dataset and their corresponding labels. These are then converted into an array of matrices where each matrix represents an image. Then this array of matrices is passed into the convolution layer. The convolution layer is fed the converted dataset. The convolution layers use 3x3 filters. The pooling layers each look at a smaller version of the image. The data representing the image is flattened and passed into the hidden layer. The first hidden layer has 128 total nodes but using random dropout half of these nodes are not used every time a batch is trained. Each subsequent hidden layer has a number of nodes that decreased by base 2 of the previous layer until the final layer is reached which has 4 nodes, which represent coordinates of the predicted box. The output layer uses softmax to give a probability that there is a pedestrian for each detected object.

In its current state, the CNN trains and saves the model, but the accuracy is not as high as desired. This is partly due to the fact that few images were used and fewer epochs were run to test functionality over accuracy. Another reason for this low accuracy is because there is currently only one convolution layer. The number of convolution layers will be increased before the final version. This will be done by adding a layer and testing accuracy and assessing the need for the additional layer. A separate program to test any image fed into it is still required.

```
model = tf.keras.models.Sequential([
    # Initial Convolution Stage
    tf.keras.layers.Conv2D(input_shape=(224,224,3), filters=64, kernel_size=(3,3), padding="same", activation="relu"),

    # Convolution layer 1 - input shape will be determined from previous output
    tf.keras.layers.Conv2D(filters=64, kernel_size=(3,3), padding="same", activation="relu"),

    # Apply max pooling on previous layer
    tf.keras.layers.MaxPool2D(pool_size=(2,2), strides=(2,2)),

    # Convolution layer 2
    tf.keras.layers.Conv2D(filters=128, kernel_size=(3,3), padding="same", activation="relu"),

    # Convolution layer 2
    tf.keras.layers.Conv2D(filters=128, kernel_size=(3,3), padding="same", activation="relu"),

    # Apply max pooling on previous layers
    tf.keras.layers.MaxPool2D(pool_size=(2,2), strides=(2,2)),

    # flatten layers
    tf.keras.layers.Flatten(),

    # add a Dense layer
    tf.keras.layers.Dense(128, activation="relu"),

    # add dense layer with half units are previous
    tf.keras.layers.Dense(64, activation="relu"),

    # keep adding dense layers until 4 units
    tf.keras.layers.Dense(32, activation="relu"),

    # final layer with 4 units - corresponds to each coordinate box
    tf.keras.layers.Dense(4, activation="sigmoid")
])
```

Figure 4 Initial CNN (Code)

Results

The results from this method show that this method, in its current state, should not be used for pedestrian detection. There are a few reasons that these results are not satisfactory. First the dataset that was used did not have a variety of images and the number of images did not allow the neural network to accurately make connections and find patterns in pedestrians. This method also is meant to serve as a base for a larger system so the accuracy is expected to be lower. If this neural network was used as a base with a region proposal network then the accuracy and time to train would greatly improve. A region proposal network allows images to be split into regions. Each of these regions would then be passed into the neural network. This is done by using a selective search algorithm which uses a sliding box that goes through the image from left to right and top to bottom similar to a convolution layer. After going through the entire image, it returns a set number of bounding boxes in random sections of the image. These bounding boxes can then be used one at a time to find the intersection with the original labeled bounding box from the dataset.

Some example outputs from the neural network are shown below in figures 5 and 6.

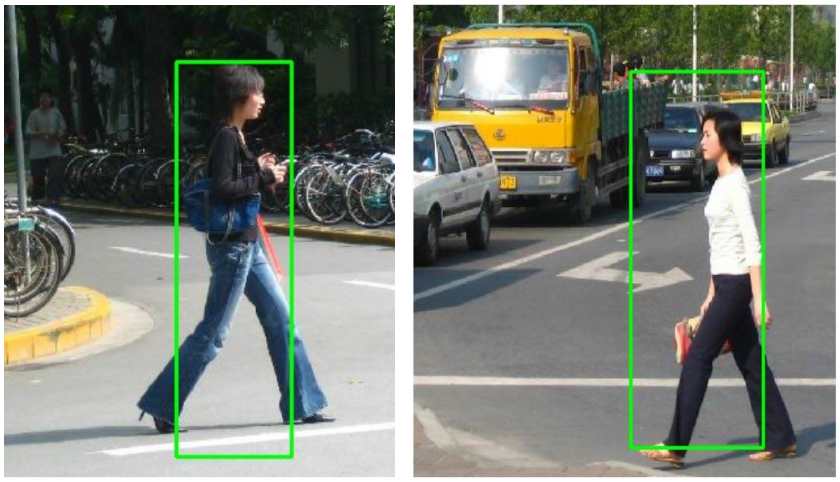

Figure 5

Figure 5 is an example of the neural network correctly predicting the location of the pedestrians in the images. The neural network successfully identifies pedestrians when there is only one person in the image and the pedestrian is in clear view.

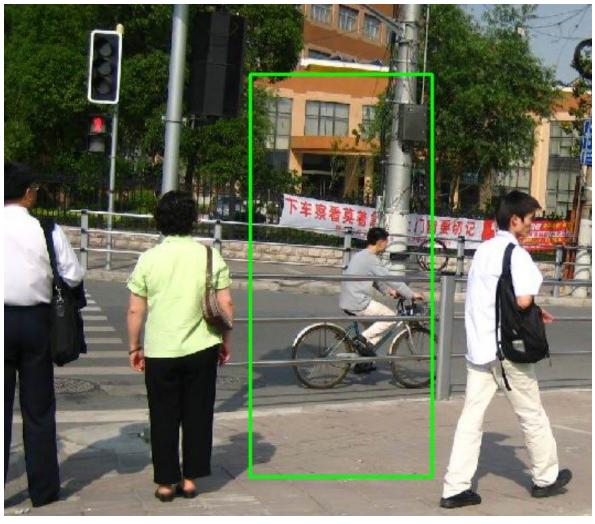

Figure 6

Figure 6 shows the neural network incorrectly identifying pedestrians in images. The neutral network mainly failed when the picture contained more than one pedestrian or there were pedestrians not in the focus of the picture. One potential way to resolve this is by adding a region proposal network so that instead of trying to identify the pedestrian in the image the network looks through the image for pedestrians in every section.

Figure 7

Figure 7 is the representation of how well the model learns per number of epochs. With small epoch values the loss is very high and the model trains the loss decreases and levels off. This is what is expected, as the model starts with random weights the learning is going to be low but as the number of epochs increases the weights are fine tuned until the resulting loss is reasonable. The above graph represents the change in mean squared error as the training progresses.

Conclusion

Pedestrian detection is one of the most important factors needed for autonomous vehicles. Since this is such a vital part of autonomous vehicles it has been done in many different ways by other groups. Most of these groups start with some kind of convolution neural network and build off of it to increase speed and accuracy. We did something similar where we start by making a convolution neural network from scratch and use it as a base for pedestrian recognition.

Our approach shows that a neural network can be used to accurately identify pedestrians. We used the Penn Fudan dataset which is a common dataset used for pedestrian detection. We did modify this dataset by converting from the PASCAL format to CSV and we only used some of the images found in the dataset. These modified images were then passed into our neural network to identify pedestrians.

The final output of this neural network is a processed image that will identify any pedestrians found. Every pedestrian will be marked by a green bounding box. This is done by passing the image into the trained convolutional neural network, and the output tells whether or not there is a pedestrian found.

After considering the results of this project, it can be concluded that this is a satisfactory pedestrian detection system after some improvements have been made. Even though it can be satisfactory, there are better approaches that have already been implemented that should be used instead. In order for this approach to be on par with these other approaches a larger more varied dataset has to be used. For this network, there were no images that contained zero pedestrians and most of the images had the pedestrians in clear view with no obstructions. Our dataset also did not contain many images since our goal was to create a good base which requires many different runs with training and testing. The dataset was kept small in order to support these multiple runs but a larger more varied dataset would improve the results of our neural network. Even with further improvements, this system cannot be recommended over other approaches such as YOLOv4 [9]. These other approaches also build off of their neural network by using some kind region proposal network to process the image to identify objects with better accuracy. For our approach if a region proposal network was added then the accuracy and speed should be able to match these other methods. This was attempted but there was not enough time to complete addition to the pedestrian recognition system. This would be a good example of how our model can be used and improved.

References

[1]Chi, Cheng, Shifeng Zhang, Junliang Xing, Zhen Lei, Stan Z. Li, and Xudong Zou. "PedHunter: Occlusion Robust Pedestrian Detector in Crowded Scenes." In *AAAI*, pp. 10639-10646. 2020.

[2]Dong, Guo, and Ming Xie. "Color clustering and learning for image segmentation based on neural networks." *IEEE transactions on neural networks* 16, no. 4 (2005): 925-936.

[3]He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision* (pp. 2961-2969).

[4]H. Zhang, Y. Du, S. Ning, Y. Zhang, S. Yang, and C. Du, "Pedestrian Detection Method Based on Faster R-CNN," 2017 13th International Conference on Computational Intelligence and Security (CIS), Hong Kong, 2017, pp. 427-430, doi: 10.1109/CIS.2017.00099.

[5]J. Duan, L. Shi, J. Yao, D. Liu, and Q. Tian, "Obstacle detection research based on four-line laser radar in-vehicle," 2013 IEEE International Conference on Robotics and Biomimetics (ROBIO), Shenzhen, 2013, pp. 2452-2457, doi: 10.1109/ROBIO.2013.6739839.

[6] "Open Images V6 - Now Featuring Localized Narratives," 26-Feb-2020. [Online]. Available: https://ai.googleblog.com/2020/02/open-images-v6-now-featuring-localized.html. [Accessed: 08-Oct-2020].

[7]R. Dass, Priyanka, and S. Devi, "Image Segmentation Techniques 1," *CiteSeerX*, Jan-2012. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.227.6638. [Accessed: 22-Sep-2020].

[8]R. Kulkarni, S. Dhavalikar and S. Bangar, "Traffic Light Detection and Recognition for Self Driving Cars Using Deep Learning," 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA), Pune, India, 2018, pp. 1-4, doi: 10.1109/ICCUBEA.2018.8697819.

[9]S. Lin, M. Lin, Y. Hwang, and C. Fan, "Deep-Learning Based Pedestrian Direction Detection for Anti-collision of Intelligent Self-propelled Vehicles," 2019 IEEE 8th Global Conference on Consumer Electronics (GCCE), Osaka, Japan, 2019, pp. 387-388, doi: 10.1109/GCCE46687.2019.9015528.

[10]T. Liu and T. Stathaki, "Faster R-CNN for Robust Pedestrian Detection Using Semantic Segmentation Network," *Frontiers in Neurorobotics,* 2018. Available: http://ulib.iupui.edu/cgi-bin/proxy.pl?url=http://search.proquest.com.proxy.ulib.uits.iu.edu/docview/2294004610?accountid=7398. DOI: http://dx.doi.org.proxy.ulib.uits.iu.edu/10.3389/fnbot.2018.00064.

[11]Z. Ji, W. Zheng, and Y. Pang, "Deep pedestrian attribute recognition based on LSTM," 2017 IEEE International Conference on Image Processing (ICIP), Beijing, 2017, pp. 151-155, doi: 10.1109/ICIP.2017.8296261.

[12]https://www.cis.upenn.edu/~jshi/ped_html/