Pedestrian Detection System for Smart Communities Using Deep Convolutional Neural Networks

Jonathan Lwowski, Prasanna Kolar, Patrick Benavidez, Paul Rad, John J. Prevost and Mo Jamshidi
Department of Electrical and Computer Engineering
The University of Texas at San Antonio
One UTSA Circle, San Antonio, TX 78249, USA
Email: jonathan.lwowski@my.utsa.edu, prasanna.kolar@utsa.edu, patrick.benavidez@utsa.edu, peyman.najafirad@utsa.edu, jeff.prevost@utsa.edu, moj@wacong.org

Abstract—Pedestrian recognition is a key problem for a number of application domains namely autonomous driving, search and rescue, surveillance and robotics. Real-time pedestrian recognition entails determining if a pedestrian is in an image frame. State-of-art pedestrian detection convolution neural networks(CNN) such as Fast R-CNN depend on computationally expensive region detection algorithms to hypothesize pedestrian locations. This paper presents a simple, fast and very accurate approach by cascading fast regional detection and deep convolution networks. Convolution networks have been shown to excel at image classification. However, convolution networks are notoriously slow at inference time. In this work, we introduce a fast regional detection cascaded with deep convolution networks that enables real-time pedestrian detection that could be used to alert a driver if a pedestrian is on the roadway. The classification CNN given an accuracy of 95.7\%, with a processing rate of about 15 frames per second on a low performance system without a graphical processing unit (GPU).

Index Terms—Deep Learning, Convolutional Neural Networks, Sliding Window, Internet of Things, Smart City, Smart Community

I. INTRODUCTION

A recent news article in San Antonio, TX stated the number of pedestrians hit by vehicles is on the rise due to one of many factors including damaged sidewalks, poor lighting, distracted drivers and speeding [1]. This is likely the case across other large cities in the country and the world. One way to help prevent these types of accidents is to develop a pedestrian detection system for alerting a driver that people are on or near the roadway.

Pedestrian detection has been prevalent in many areas such as robotics, surveillance, driver assistance systems, and autonomous transport systems. These systems can rely on camera, laser and/or radar based sensing systems. An optimal pedestrian detection system will output at an optimal data rate with high accuracy and will consume minimal power. Laser and radar based systems can be prohibitively expensive to implement for the typical consumer budget. Existing pedestrian detection systems focus either on high speed or high accuracy but not both. The problem is finding the right balance of fast processing and high accuracy.

Machine learning has proven to be the solution for many systems due to its relatively high speed and accuracy of processing data. Some of the applications Optical character recognition, include Intelligent Simultaneous Localization and Mapping(SLAM), Smart city initiatives, Image recognition, smart agriculture, to name a few [2]–[4]. There are several techniques for pedestrian detection in the literature [5]–[11]. Piot et al. performed extensive research on various algorithms [12]. Based on Piot’s research, the most accurate algorithm by Viola and Jones had 95% accuracy at 0.447 frames per second on the INRIA dataset. Viola and Jones used patterns of motion and appearances to detect pedestrians [9]. Their algorithm uses multiple frames of a video, along with AdaBoost to gain information about both the motion and appearance of the pedestrians. Although this method has a very low false positive rate, it runs at five frames per second. They demonstrate a very efficient image motion detection and a pedestrian detection system that identifies pedestrians even on low resolution images, under sub-optimal conditions of snow and rain. Others like Dalal et al used histograms oriented gradients(HOG) to achieve [13]. Wu, Bo et. al. provided results on detecting multiple, partially occluded humans in a single image by using a bayesian combination of edgelet detectors [14].

The fastest algorithm on the INRIA dataset, developed by Dollar et. al., operated at 6.49 frames per second, but only had an accuracy of 56% [15] [12]. Their algorithm, Fastest Pedestrian Detector in the West, uses a fast method for approximating features at multiple scales using a sparsely sampled image pyramid with a step size of an entire octave and within each octave we use a classifier pyramid. Their method achieves almost the same accuracy as a dense image pyramid, with the speed of a classifier pyramid.

Another challenge with pedestrian detection, is the detection and tracking of multiple people in cluttered scenes. Andriluka et. al. addressed this challenging problem by combining both detection and tracking into a single framework. Their system detects the articulation of the people in each frame and combines it with the prior knowledge of possible articulations. This is then passed into a hierarchical Gaussian process latent
variable model. This model allows the system to detect and track multiple people at a time.

Leibe et al. also researched pedestrian detection, but they looked at cases where the pedestrians had severe overlap [6], [16], [17]. They combine local and global information via probabilistic top-down segmentation. This method allows pedestrians to be detected in situations where there are severe overlaps, such as a crowd.

Google recently developed a real-time pedestrian detection system that implements a cascading deep neural network [5]. Google’s system contains a robust pre-training phase. This pre-training phase uses weights that are initialized from the weights of a network that has been trained on Imagenet. Google’s system runs at approximately 15 frame per seconds with an average miss rate of 26.2% on the Caltech Pedestrian Detection Benchmark.

In this paper we propose a deep network based system for pedestrian classification and detection. This paper focuses on improvements towards the task of pedestrian classification. Additionally, we present an initial prototype of a detection system to provide accurate location data for a person in an image frame.

The rest of the paper is organized as follows. Section II will contain a discussion of the dataset that was used. The system architecture will be presented in Section III, followed by the detection algorithm in Section IV. The concluding remarks will be discussed in the final section.

II. DATASET

The dataset used is the INRIA Person dataset. This dataset contains images from the following sources.

- GRAZ 01
- Personal digital image collections
- Google Images

This dataset comprises of training and test images that contain pedestrians and ones that do not contain pedestrians and it assists in classifying and identifying the pedestrians in the images. The training dataset has 2416 images of pedestrians and 1218 images of non-pedestrians. The testing dataset has 1132 images of pedestrians and 453 images of non-pedestrians. These images are all cropped to 64x128 pixels.

III. CLASSIFICATION ARCHITECTURE

The proposed classification architecture is broken down into several parts. These parts include pre-training processing, training, and trained model evaluation.

A. Pre-training Processing

In order to make the training phase more accurate, the dataset training images were randomized. The randomization of training images insures that the training process does not get to many pedestrian or non-pedestrian images in a row. This increases the accuracy and speed of the training. Also to simplify the model, the images were all zero padded to make them all 128x128 pixels. By having a square image, the system architecture could be simplified and easily matched to existing models.

B. Training

To begin the training process, the pre-processed training data is split into batches of 256 images. The batch of images is then passed into the convolutional neural network (CNN), as seen in Figure 1, that was developed by Angelova et al [5]. The label predicted by the model is compared to the actual label. Then using an Adaptive Moment Estimation Optimizer, the weights off the CNN are updated to try an minimize the error. After training to model for 1500 batches, the model needs to be tested on the testing dataset.

C. Trained Model Evaluation

To verify that the CNN developed by Angelova et al is trained properly, the CNN is tested on the testing dataset [5]. After training the model, the accuracy on the testing dataset was 92.8%. In order to improve the results, the outputs of the convolutional layers were analyzed. After the analysis, some of the convolutional layers’ depths were increased and an additional convolutional layer was added to the CNN, seen in figure 2. After retraining the new model, the model was again tested using the testing data. This model had an accuracy of 93.5%.

1) Analyzing Model Error: In order to try to increase the accuracy of the model, the convolutional layers were analyzed first. The outputs of the convolutional layers can be seen in Figure 3. Since the convolutional layers had no obvious flaws, the incorrect predictions were analyzed next.

There are two different kinds of incorrect predictions. There are false positives, seen in Figure 4, which is when the model predicts that an image is a pedestrian when it is not a pedestrian. There are also false negatives, seen in Figure 5, which is when the model predicts that an image is not a pedestrian when it is actually a pedestrian. Based on the fact that our model has an equal amount of false positives and false negatives, the dataset needed to be increased in size. To do this, all of the images were flipped horizontally. This creates double the number of images as compared to the original dataset.

After retraining the model for 1900 epochs, the new model had an accuracy of 94.5%.

After analyzing the incorrectly predicted images again, several things were noticed. The images that the model incorrectly predicted were abnormal cases, such as a woman carrying an umbrella and images with low lighting. A few examples of these incorrectly predicted images can be seen in Figure 6. To solve this problem, Gaussian noise was added to the incorrectly identified images. The new images were then added back into the dataset. The model was then trained for an additional 1100 epochs and the new accuracy was 95.7%.

Now that the classification CNN has a very high accuracy, the sliding window detection algorithm can be added on top of the CNN to calculate the exact locations of pedestrians in the camera frame.

IV. DETECTION ALGORITHM

There are many classical methods for the detection of objects in an image, such as dense image pyramids and classifier
Fig. 1: Initial convolutional neural network model used for pedestrian classification.

Fig. 2: Final convolutional neural network model used for pedestrian classification.

Fig. 3: Outputs of the convolutional layers

pyramids [18]. There are also many different feature detection methods such as fast feature pyramids that can quickly calculate places in the image where there could potentially be a person [18]. Since the focus of this paper is not the detection algorithm, a slower but easier to implement sliding window approach was used. This algorithm is described below.

A. Feature Detection Algorithm

Given a positive identification of a person in an image, it is important to determine where exactly the person is in the picture. We make the following assumptions to identify pixels in an image that most likely contain a person. The
first assumption is that a person will appear different in color content than the background scenery. The second assumption is that a person will likely be far from the camera. These simple assumptions provide a basis for further examination of pixel content that is dissimilar to a majority of the image. To perform the comparison, a computationally inexpensive color transform called rg-chromaticity is utilized. This color transform excels at highlighting the relative ratios of red, green and blue channels to the picture in a variety of lighting conditions. Equations 1, 2 and 3 provide the per-pixel calculations necessary to perform the color space transformation, where $R$, $G$, $B$ represent the intensity of the color in a pixel, and $r$, $g$, $b$ represent the proportion of the single channel to the overall color of the pixel.

$$r = \frac{R}{R + G + B}$$  \hspace{1cm} (1)

$$g = \frac{G}{R + G + B}$$  \hspace{1cm} (2)

$$b = 1 - r - g$$  \hspace{1cm} (3)

The color transformation is used in developing the image mask, which indicates the pixels where a person can be found. Equation 4 provides the general calculation of the image mask

$$mask = (r_{min} < r < r_{max}) \cap (g_{min} < g < g_{max})$$  \hspace{1cm} (4)

where $r_{min}$ and $r_{max}$ indicate the lower and upper bounds of the relative red intensities, and $g_{min}$ and $g_{max}$ indicate the lower and upper bounds of the relative green intensities. Figure
8 shows the detected points from Figure 7 using the range values $r_{\text{min}} = 0.43$, $r_{\text{max}} = 1.0$, $g_{\text{min}} = 0.0$, and $g_{\text{max}} = 0.25$. While the criteria is statically selected and relatively simple to implement, the reduction in the number of points to process is substantial. The selection criteria can be further improved dynamically by taking the histogram and finding the most relevant non-background pixels.

**B. Sliding Window Algorithm**

Once the features have been detected, a sliding window algorithm is used. The sliding window algorithm starts with a small window that is slid to the detected feature points as seen in Figure 9. Each window is passed through the CNN for classification. If the CNN classifies the window as a pedestrian, a bounding box is put around the window. Once the window has been slid across the entire image, the window’s size is increased. The same classification process is used until the window becomes large. Once the sliding window process is finished, all the people in the image will have a bunch of overlapping bounding boxes around them. This can be seen in Figure 10.

**C. Removing Overlapping Bounding Boxes**

To remove the overlapping bounding boxes, a simple algorithm is used. The algorithm checks every bounding box, and if that bounding box is overlapping another bounding box, the two boxes are combined into one larger box. This process is repeated until there are no more boxes overlapping. The results of the overlapping bounding box removing algorithm can be seen in Figure 11. Now all of the centers of the bounding boxes are the locations of people in the image.
D. Detection Algorithm Results

Using the classical version of the sliding window approach in which every pixel in the image is visited, the average runtime of the algorithm was 14874 seconds. In comparison our sliding window approach combined with the feature detection algorithm only had an average runtime of 96 seconds.

V. Conclusions

Pedestrian detection systems detect standing or walking humans and machine learning is a concept when implemented in computing systems can be trained to identify objects and in this case the object of identification is a pedestrian. This coupled system of a pedestrian detection with machine learning concepts is useful when implemented on autonomous mobile systems like cars, unmanned aerial systems etc. This is because once we have a well trained system, machine learning systems are portable, accurate and fast. The proposed classification system has been designed to be simple, yet fast and accurate at classifying pedestrians. Presently, the system has been implemented on a cloud based system and a bare metal GPU system with a classification accuracy above 95.7%. The detection system operates at an average detection speed of 96 seconds. In the future, a fast feature detection will be designed to detect the places in the image where a pedestrian could be located. This algorithm will allow the sliding window approach to be sped up to reasonable run-times. Once the detection algorithm is sped up, this system can be implemented on mobile systems for example a public transport bus, to identify pedestrians in the vicinity and avoid accidents.

Acknowledgments

The authors would like to thank everyone in the Autonomous Control Engineering Lab for their support and help in this project. This work was supported by Grant number FA8750-15-2-0116 from Air Force Research Laboratory and OSD through a contract with North Carolina Agricultural and Technical State University.

References