International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 07 Issue: 08 | Aug 2020 www.irjet.net p-ISSN: 2395-0072

Baby Monitoring System using Object Detection

Arjun S Kaushik[1], Soorya Raysam[2], Venkatakrishna S[3], Dr Prabhanjan S[4]

[1,2,3]*Dept. of Computer Science, Jyothy Institute of Technology, Bangalore, India*
[4]*Head of Department, Dept. of Computer Science, Jyothy Institute of Technology, Bangalore, India*

Abstract - *In practice, there are two types of mainstream object detection algorithms. Algorithms like R-CNN and Fast(er) R-CNN use a two-step approach - first to identify regions where objects are expected to be found and then detect objects only in those regions using convnet. On the other hand, algorithms like YOLO (You Only Look Once) and SSD (Single-Shot Detector) use a fully convolutional approach in which the network is able to find all objects within an image in one pass (hence 'single-shot' or 'look once') through the convolutional network. The region proposal algorithms usually have slightly better accuracy but slower to run, while single-shot algorithms are more efficient and have as good accuracy.*

The objective of this project is to analyse and detect the presence of a baby in a real-time video feed. This is achieved through the use of SSD (Single-Shot Detector). If the baby is found within the vicinity of the frame, the real-time video feed is just displayed to the registered user. If the baby is missing from the frame for more than a certain duration of time, a notification system which exists, is alerted. This leads that system to send a notification to the registered user regarding the absence of the baby. Thus, on receiving the alert message, the parent can take appropriate action.

Keywords: **Convolutional Neural Network, Object Detection, Single Shot Detection, YOLO, Tkinter** .

INTRODUCTION

1.1 SSD (Single Shot Multibox Detector):

Nowadays, deep learning networks are better at image classification than humans, which shows just how powerful this technique is. However, we as humans do far more than just classify images when observing and interacting with the world. We also localize and classify each element within our field of view. These are much more complex tasks which machines are still struggling to perform as well as humans. The name of this architecture comes from:

- Single Shot: this means that the tasks of object localization and classification are done in a single forward pass of the network
- MultiBox: this is the name of a technique for bounding box regression.
- Detector: The network is an object detector that also classifies those detected objects

MultiBox

The bounding box regression technique of SSD is inspired by Szegedy's work on MultiBox a method for fast class-agnostic bounding box coordinate proposals. Interestingly, in the work done on MultiBox an Inception-style convolutional network is used. The 1x1 convolutions that you see below help in dimensionality reduction since the number of dimensions will go down (but "width" and "height" will remain the same).

MultiBox's loss function also combined two critical components that made their way into SSD:

Confidence Loss: this measures how confident the network is of the objectness of the computed bounding box. Categorical cross-entropy is used to compute this loss.

Location Loss: this measures how far away the network's predicted bounding boxes are from the ground truth ones from the training set.

1.2 TensorFlow

TensorFlow is an open source library for numerical computation and large-scale machine learning. TensorFlow bundles together a slew of machine learning and deep learning (aka neural networking) models and algorithms and makes them useful by way of a common metaphor. It uses Python to provide a convenient front-end API for building applications with the framework

In Tensorflow, all the computations involve tensors. A tensor is a vector or matrix of n-dimensions that represents all types of data. All values in a tensor hold identical data type with a known (or partially known) shape. The shape of the data is the dimensionality of the matrix or array.

A tensor can be originated from the input data or the result of a computation. In TensorFlow, all the operations are conducted inside a graph. The graph is a set of computation that takes place successively. Each operation is called an op node and are connected to each other. The graph outlines the ops and connections between the nodes. However, it does not display the values. The edge of the nodes is the tensor, i.e., a way to populate the operation with data.

2. LITERATURE SURVEY

The authors present a method for detecting objects in images using a single deep neural network. The approach, named SSD, discretizes the output space of bounding boxes into a set of default boxes over different aspect ratios and scales per feature map location. At prediction time, the network generates scores for the presence of each object category in each default box and produces adjustments to the box to better match the object shape. Additionally, the network combines predictions from multiple feature maps with different resolutions to naturally handle objects of various sizes. SSD is simple relative to methods that require object proposals because it completely eliminates proposal generation and subsequent pixel or feature resampling stages and encapsulates all computation in a single network. This makes SSD easy to train and straightforward to integrate into systems that require a detection component. Experimental results on the PASCAL VOC, COCO, and ILSVRC datasets confirm that SSD has competitive accuracy to methods that utilize an additional object proposal step and is much faster, while providing a unified framework for both training and inference. For 300 × 300 input, SSD achieves 74.3% mAP1 on VOC2007 test at 59 FPS on a Nvidia Titan X and for 512 × 512 input, SSD achieves 76.9% mAP, outperforming a comparable state-of-the-art Faster R-CNN model. Compared to other single stage methods, SSD has much better accuracy even with a smaller input image size.[1]

Pedestrian detection is a valuable and challenging problem in computer vision. To fully exploit the interframe information to improve the detector's performance, many frameworks with high complexity for offline detection have been proposed. These methods cannot provide spontaneous responses or alerts. In this paper, we present a Kalman filter-based convolutional neural network (CNN) for online pedestrian detection in videos. First, the single shot multibox detector is implemented as the CNN detector, which incorporates the pedestrian's aspect ratios. Fusion modules are implemented to improve the detector's robustness for medium and far scale pedestrians. Then, bounding boxes are propagated according to the prediction from the Kalman filter. Finally, the location and confidence of the bounding boxes are refined by the Kalman filter. This method is evaluated on two datasets with respect to both the miss rate and speed, and the results show that this method has a lower miss rate, more stable confidence, and a much higher speed. [2]

A method of multi-block Single Shot MultiBox Detector (SSD) based on small object detection is proposed to the railway scene of unmanned aerial vehicle surveillance. To address the limitation of small object detection, a multi-block SSD mechanism, which consists of threesteps, is designed. First, the original input images are segmented into several overlapped patches. Second, each patch is separately fed into an SSD to detect the objects. Third, the patches are merged together through two stages. In the first stage, the truncated object of the sub-layer detection result is spliced. In the second stage, a sub-layer suppression and filtering algorithm applying the concept of non-maximum suppression is utilized to remove the overlapped boxes of sub-layers. The boxes that are not detected in the main-layer are retained. In addition, no sufficient labeled training sam-ples of railway circumstance are available, thereby hindering the deployment of SSD. A two-stage training strategy leveraging to transfer learning is adopted to solve this issue. The deep learning model is preliminarily trained using labeled data of numerous auxiliaries, and then it is refined using only a few samples of railway scene.[3]

3. REQUIREMENTS ANALYSIS

3.1 Hardware Requirements

Processor: Any Processor above 500 MHz

Ram: 8 GB

Hard Disk: 4 GB

Dedicated Graphics Card

Input device: Standard Keyboard, Mouse and Camera.

Output device: High Resolution Monitor.

3.2 Software Requirements:

Operating System: Windows 7 or higher

Programming language: Python and related libraries.

Software: Anaconda Version 3.6

4. IMPLEMEMTATION

4.1 Object Detection Model

An object detection model is trained to detect the presence and location of multiple classes of objects. For example, a model might be trained with images that contain various pieces of fruit, along with a *label* that specifies the class of fruit they represent .Given an image or a video stream, an object detection model can identify which of a known set of objects might be present and provide information about their positions within the image.

1. Dataset:

A custom dataset was used for training the object detection model. We have used a dataset containing about 500 samples with two classes(baby,adult).

2. Model Training

The data set was divided into training and testing with training ratio 7:3 .The model was trained by using SSD as the pre-trained model, with tensorflow object detection api.

3. Real time testing:

After the training of proposed object detection model, the trained model was tested in real time. First, a person was detected by the computer camera. After that, the detected images were sent to the model and the classes they belong to were queried. As a result of the predictions, the possibility of belonging to which class, among the two classes was shown on the screen.

OpenCV is used to draw a rectangular boundary around the detected person and the resulting class is displayed on the screen with the percentage confidence.

4.2 Notification Service

A **notification service** provides means to send a notice to many persons at once.

The notification service used here is called notifiy-run. This is a simple python based notification service where a notification channel is created and the users can join this channel by their devices to which the notification can be sent.

4.3 Streaming Service

A video streaming server delivers live video or a recorded content over the Internet or through the local network to a user with a computer, smartphone, or other connected device. The term "streaming" refers to the actual process of transmitting the video, with the server in a constant state of delivering the content to the devices.

The streaming service used in this project is called Flask-OpenCV-Streamer. Flask is a micro web framework developed using python. Flask is basically used for building lightweight web applications. Using this framework the OpenCV footage can be easily streamed over the local network.

4.4 Baby Monitoring Application

Desktop Applications are stand alone applications which are used on the user's laptops and systems. The desktop application built in this project integrates all the above mentioned services and works synchronously.

Building the application:

The application is implemented using Tkinter and other python functions. Tkinter is one of the most frequently used GUI toolkits. There are various python libraries which are used to implement radio-buttons, buttons, and text boxes. The main functions of this application are starting the detection service along with the notification service and the streaming server and running it synchronously. The application has a separate section for handling notification functions such as creating a notification channel only on the first run of the application and for adding new devices for the existing channel by means of qr code or through registration link.

The list of various functions used in the code is as follows:

Function Name	Description
closewin()	Destruction of windows
window()	Creation of window and setting orientation of the window
detect()	Function for running the monitoring service
notifywin()	Function for handling notification services
add_device()	Function for adding new devices to the notification channel
new_reg()	Function for creating a new notification channel
qr()	Function for creating a qr code for registration
clipboardc(link)	Function for copying the registration link to clipboard
checknet()	Function for checking connectivity status

Table 4.4.1 Functions used

Analysis:

The system checks if the baby is present in the given video feed. If the baby is not present or if the system is unable to detect the baby, the system waits for around 5 seconds before determining that the baby has left the area or the room. This is done in order the eliminate detection error, i.e in order to prevent the system from sending false notification. During this wait period if the system is able to detect the baby again then the counter which is responsible for the waiting period is reset .If the system is unable to detect the baby after the waiting period, a notification is sent to the registered devices stating that the baby is not found. Upon interacting with the notification(clicking or tapping the notification),the device opens the live video feed from the camera and displays it on the device screen for the user in order to physically confirm or to check if the baby is present or not. Before sending the notification, the system makes sure that the internet connection is available so that the system does not wait for the notification service to complete its

execution. This is done in order to eliminate dead locks where the system keeps on waiting for the notification service to complete its task, when there is no internet connection. The user also has the option to explicitly enable or disable the notification service from the desktop application.

5. RESULTS AND DISCUSSIONS

We trained our Object detection model using a custom dataset consisting of two classes (baby, adult).The output of the model is used by the application in order to determine the presence of the baby.

In this paper, we introduced a desktop application to detect whether the baby is present or not and depending upon that it sends a notification to the parent/guardian.

Figure 5.1 Baby Detection

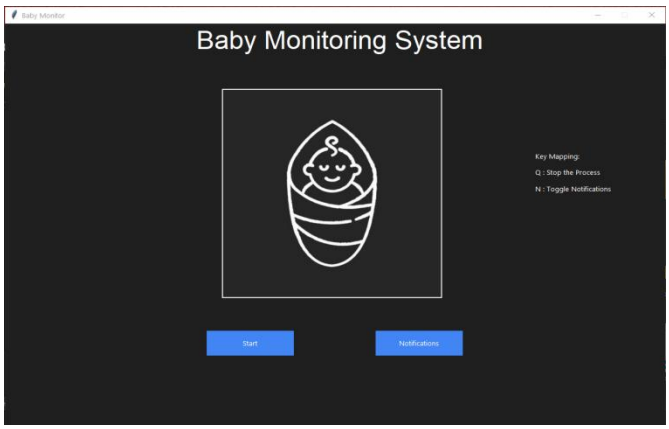

Figure 5.2 Application implementation

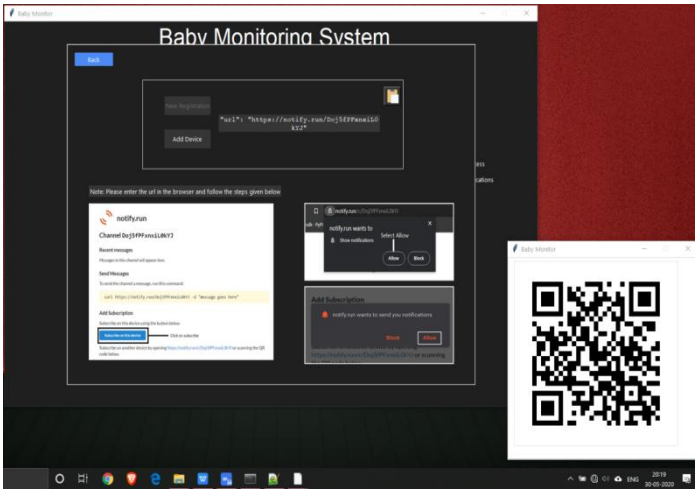

Figure 5.3 Notification Registration

6. CONCLUSION

An automated Baby monitoring System can be used in most of the houses since the proposed system does not require any extra initial investment or extra devices such as an actual CCTV. This system can made using an old smartphone or a camera and an existing PC. The system has 4 modules-baby detection that is implemented by using SSD model, notification service implemented using notify-run, streaming service that streams the video over a local network. Finally, the last module, a desktop application that is used to run all these modules synchronously and used to interact with these services.

Due to its efficiency or ease of implantation the above stated algorithms are selected for the proposed system.

7. FUTURE SCOPE

The project mainly deals with baby monitoring using a desktop based application

As of the current working, the desktop application is local, except for the notification service. In the future days, this application can be hosted on a website using an internet connection.

The current system uses a single camera ,in the future versions this system can be implemented using stereo cameras ,which can be used to estimate the distance between two objects .Using this method the application can be developed to detect if the baby is about to exit the room in a more efficient way .

REFERENCES

1. Wei Liu1, Dragomir Anguelov2, Dumitru Erhan3, Christian Szegedy3,Scott Reed, Cheng-Yang Fu1, Alexander C. Berg, “SSD: Single Shot Multi Detector”

2. R. FAN YANG 1, HOUJIN CHEN1, JUPENG LI1, FENG LI2, LEI WANG1, XIAOMIAO YAN, "Single Shot Multibox Detector With Kalman Filter for Online Pedestrian Detection in Video"

3. Yundong LI,Han Dong ,Xueyan ZHANG ,Baochang ZHANG ,Zhifeng XIAO ,Hongguang LI , "Multi-block SSD based on small object detection for UAV railway scene surveillance".

4. M. Ozuysal, P.Fua and V. Lepetit "Fast key porint recognition in Ten Lines of Code".

5. P. Viola and M.Jones "Robust Real-time Object Detection".

6. R. Duraiswami and H. Samet "SoftPOSIT: Simultaneous Pose and Correspondence Determination".

7. P. Fua and V.Lepetit "A Fast Local Descriptor for Dense Matching".

8. B. Zitova and J.Flusser "Image registration Methods:a Survey".

9. Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Network".

10. Ronald and S. Zimmermanna, Julien N. Siemsa "Faster Training of Mask R-CNN by Focusing on Instance Boundaries.

11. Christian Szegedy, Alexander Toshev,Dumitru Erhan "Deep Neural Network for Object Detection".