Abstract—This paper proposes an edge camera system using a compressed deep learning model for enhanced Video surveillance Management System of Smart City that analyzes existing recorded video. The proposed edge camera at the end terminal of the Video Surveillance Management System send the analyzed image, information and warning to the central system according to the situation analysis based on the information obtained by compressed deep learning for low memory and real-time operation in embedded system. We tested with edge camera installed in street lamp and confirmed stable operating performance and high recognition rate compared to existing system.

I. INTRODUCTION

Recently, due to the population densification in the city, various incidents such as crime are occurring in various places such as a street or an alley. As a result, the installation of video surveillance systems greatly expand to prevent crime and accidents and to provide traffic surveillance, facility protection, and disaster management [1-3]. The spread about that system is increasing due to the performance of intelligent image analysis system been improved through the many researches on detection technology using deep learning based AI. The existing video security system that has been recording or monitoring only in case of emergency is contributing to the safety of people in various fields such as firefighting, defense and transportation through the improved intelligent image analysis technology.

A typical video surveillance management system can divide into a centralized surveillance system and a distributed surveillance system based on edge cameras. Among them, the centralized surveillance system mainly used that monitors all images in the control center is constructed and because of privacy and security problems. This system stores images on a server in the central system, records more than 16 images per server, and examines more than 50 images per supervisory workforce. Therefore, if there is a problem in the surveillance area, it used only for solving the problem after the occurrence of the problem or for specifying the fine. Recently, the new system has an intelligent image analysis module, which can analyze the images stored in the server, store various information about the image at the same time, or search for various parts of the image. In addition, due to the AI technology based on deep learning, it can accurately detection and analysis and it uses for various purposes. The intelligent image analysis system must be composed of a number of GPUs, power, etc. in order to use the deep learning. It causes excessive cost and long-term cost due to power consumption. A distributed system based on edge camera is not yet widely used, if it can operate a high-performance image analysis technique on the embedded system then it can supply various information analyzed in the edge. It have recently received much attention as a system that sends detected information to the control center and provides alarms to the surveillance workers in case of a specific problem and enables detailed monitoring afterwards[4-5].

The object detection for edge cameras used simple method to operate in embedded system. It mainly used machine learning to detect various information, but performance was slow and speed was slow. In recent years, due to the evolution of AI based on deep learning, recognition performance has improved greatly, and it studies various optimization method to operate real time in embedded system. Particularly, the reduction of deep learning studies through various method. A method to reduce the layer depth or adjusts the layer size through analysis about the network model of deep learning, and a method to reduce the model itself, and a method to convert a deep learning model based on GPU to model based on CPU[6-11]. In addition, the latest embedded system consists GPU to enable deep learning, and it develops that can utilize deep learning. NVIDIA supports deep learning with Jeston board that it can utilize CUDA, cuDNN, TensorRT. Intel supports using Movidius VPU, and TI supports similar GPU operation using DSP (Digital Signal Processor) which specialized for floating-point operation and EVE (Embedded Vision Engine) which specialized for parallel operation. Deep learning algorithms based on various GPU operations apply to embedded application. Recently, deep learning method optimized for embedded application develops [12-15].

In this paper, we propose an edge camera system using object detection method optimized for embedded system on compressed deep learning method through Hint training. We propose a model compression technique that optimizes model not to degrade a performance of object detection. We optimize the deep learning technology configured in C language to fit the embedded system, and realize real time detection in the edge camera, and we propose the operating software of embedded system for connection with the control center and edge camera system. We introduce the proposed method, show the edge camera system made using it, and show
the detection performance of the object detection method based on the compressed deep learning model.

II. EDGE CAMERA SYSTEM

A. System Overall

In this paper, the edge camera system proposed has the configuration as shown in Fig. 1. The system consists of a camera and an embedded board. It detects objects that optimized for security by configuring an object detection method based on a deep learning model. This method divided into each threads that receives camera images as inputs, detect objects, and recognize objects. It re-encoded the image which they marked the detection information and it transmitted to the Video Surveillance Management System using RTSP (Real Time Streaming Protocol), and information about the dangerous situation is provided to the central system administrator.

![Fig. 1. System Overall](image1)

B. Hardware Implementation

Although there are many researches on deep learning, there are many restrictions on deep learning based object detection method in embedded boards. Recently, various companies such as NVIDIA, Intel, and TI have released high-performance embedded boards. However, it is not satisfied with the price and performance level that can utilize in the CCTV market.

We constructed the edge camera as shown in Fig. 2 by applying the obstacle detection method for the edge camera configured in C language to NVIDIA Jetson TX2 considering the proper level and usability. NVIDIA Jetson TX2 is a promising AI SoC powered by NVIDIA Pascal GPU architecture for inference at the edge. It consists of a quad-core 2.0 GHz 64-bit ARMv8 A57 processor, a dual-core 2.0 GHz superscalar ARMv8 Denver processor, and an integrated Pascal GPU 1.3 GHz with 256 cores. The six CPU cores and the GPU share 8 GB DRAM memory [16-17].

We propose an embedded S/W for operating security-optimized edge cameras. The object detection algorithms based deep learning separated and configured to operate based on a certain time of each thread, and each thread manages memory and GPU to constantly control. As shown in Figure 1, Thread I resizes the image size to fit network size, Thread II detects obstacles based on deep learning using GPU, Thread III analyzes the detected objects and extract valid objects. The information of the finally valid objects display on the image and transmitted to the central system by RTSP. RTSP is a protocol related to real-time streaming. The transmission of streaming data itself is not a task of RTSP. Most RTSP servers use the Real-time Transport Protocol (RTP) in conjunction with Real-time Control Protocol (RTCP) for media stream delivery. When the central system accepts request of the protocol sent from edge camera, it is configured to receive the processing result through RTP. At this time, text information about the situation simultaneously transmits to alarm the central system administrator.

![Fig. 2. Prototype of the Edge Camera System](image2)

C. Detection Algorithm based on Model Compression

Deep learning based CNN (convolutional neural networks) has excellent performance for classification, detection, and segmentation about objects. However, it is suitable for systems using high performance GPUs and difficult to use for embedded systems, various research such as mobile-net, squeeze-net for mobile networks is recently underway. In this paper, it uses Hints Training to compress CNN based YOLO which detects object in real time [18-20]. The original version of YOLO made up of C language, which is highly utilized and can utilize the same weight in general PC and embedded system. The detection speed is fast, so it is easy to apply even in embedded environment with low calculation power. YOLO is version 3 recently but it uses version 2 as a reference in consideration of memory usage. In order to reduce the weight, we divided the structure of the deep learning model into sections and learned each section as shown in Fig. 3.

We compress the model using the following Hints Training in YOLO as shown in Figure 4. First, a model with reduced number of convolutional layers generated in the original model. We changed the last activation function to ReLU function to Linear function, and the cost uses the RMSE (Root Mean Square Error) value. After that, initialization carried out by
collecting the weight for each part, and then it construct an optimized lightweight model through many a fine-tuning process. In particular, we used the Linear function to keep track of the original model without any limitation on the output value.

III. EXPERIMENTS

A. Experiment Environment

We evaluated the performance of the edge camera system in various fields considering the specificity using the embedded system. Especially, for the object detection algorithm, the recognition performance, the detection speed, the operation speed, the memory quota and the temperature for the edge camera system were experimented.

Experiments on compressed deep learning model based object detection technology were conducted based calculate Precision and Recall for test set after learn about VOC dataset for 5 classes (Person, Car, Bus, Bicycle, Motorbike) required for traffic safety. The edge camera system evaluated the performance by measuring the average detection rate, the average memory usage, and the CPU or GPU average temperature when driving the actual algorithm for a long time.

B. Experiment Results

Experimental results of object detection are shown in Fig.5. The upper left in Fig. 5 shows the result using YOLOv2, the upper right shows the result using YOLOv2-tiny, and the below shows the result using the proposed algorithm. The results of the object detection show that the performance of the lightweight algorithm is not significantly lower than that of the YOLOv2 model, and it can confirm that the recall and precision is similar than YOLOv2 as shown in Table 1. In the same PC environment, the operation time of the proposed algorithm is shown in Table 2 and it is confirmed that it is a bit slower than YOLOv2-tiny although the speed is faster than YOLOv2.

![Fig. 3. Hint training for Model Compression](image)

![Fig. 4. Reduced Convolutional layer in YoloV2](image)

<table>
<thead>
<tr>
<th>TABLE 1</th>
<th>THE RESULT OF DETECTION RATIO (THRESHOLD 0.2)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>All</td>
</tr>
<tr>
<td>----------</td>
<td>-------</td>
</tr>
<tr>
<td>P</td>
<td>R</td>
</tr>
<tr>
<td>YOLOv2</td>
<td>62.8</td>
</tr>
<tr>
<td>YOLOv2-tiny</td>
<td>58.6</td>
</tr>
<tr>
<td>Proposed Algorithm</td>
<td>62.7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TABLE 2</th>
<th>THE RESULT OF DETECTION SPEED</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inference Time (1080 Ti)</td>
<td></td>
</tr>
<tr>
<td>YOLOv2</td>
<td>9.37 ms</td>
</tr>
<tr>
<td>YOLOv2-tiny</td>
<td>3.64 ms</td>
</tr>
<tr>
<td>Proposed Algorithm</td>
<td>5.36 ms</td>
</tr>
</tbody>
</table>

The experimental results for the edge camera system are shown in Fig. 6, which shows the screen transmitted through RTSP in the proposed system. Especially, the operation time is
37.98ms on average and memory uses 1.38GB of 7.85GB, which is lower than that of YOLOv2 using 2.43GB. In addition, the average temperature of CPU and GPU during operation time was 59 ~ 60 and the temperature was higher than the standard of CCTV system.

![Image](Image 47x511 to 298x659)

**Fig. 6. Image Received from Edge Camera through RTSP**

### IV. CONCLUSION

In this paper, we propose an edge camera system using an object detection algorithm based on a deep learning model based on model compression for embedded applications. The existing system that controls based on the result of analyzing images obtained a distributed camera in each region in the central system requires high computing power and cost and requires a large number of surveillance personnel. On the other hand, a distributed system that requests surveillance based on the results analyzed by each edge camera can do surveillance a larger number of cameras and areas with a few man. The proposed edge camera in this paper confirmed the same detection performance by operating in real time in the embedded system by compressing the existing object detection method. Based on the proposed method, prototype of edge camera was fabricated and its system performance was verified. Experimental results show that the proposed algorithm confirmed similar detection performance without degrading performance compared to the original model, and the edge camera system has high temperature, but it confirmed stable recognition performance, speed and low memory usage. Based on these advantages, it can apply to various fields as a security camera installed in a street lamp or a traffic light.

In the future, we will build an edge camera system using Intel and TI DSP chips as well as NVIDIA Jetson. We will study the new structure and operation S/W that solves the problem of high operating temperature compared to the existing CCTV. We will also study the new detection model by applying the model compression technique for YOLOv3, which uses more memory than YOLOv2 but has higher recognition performance.

### REFERENCE STYLES


[8] Song Han, Huizi Mao, William J. Dally, “Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding,” ICLR 2016


