

## Article

# EGCY-Net: An ELAN and GhostConv-Based YOLO Network for Stacked Packages in Logistic Systems

Indah Monisa Firdiantika , Seongryeong Lee, Chaitali Bhattacharyya , Yewon Jang and Sungho Kim \* 

Department of Electronic Engineering, Yeungnam University, 280 Daehak-ro, Gyeongsan-si 38541, Republic of Korea; indahmonisaf@yu.ac.kr (I.M.F.); gkd16986@yu.ac.kr (S.L.); bhchaitali@yu.ac.kr (C.B.); zzolimi49@yu.ac.kr (Y.J.)

\* Correspondence: sunghokim@yu.ac.kr; Tel.: +82-53-810-3530

**Abstract:** Dispatching, receiving, and transporting goods involve a large amount of manual effort. Within a logistics supply chain, a wide variety of transported goods need to be handled, recognized, and checked at many different points. Effective planning of automated guided vehicle (AGV) transportation can reduce equipment energy consumption and shorten task completion time. As the need for efficient warehouse logistics has increased in manufacturing systems, the use of AGVs has also increased to reduce working time. These processes hold automation potential, which we can exploit by using computer vision techniques. We propose a method for the complete automation of box recognition, covering both the types and quantities of boxes. To do this, an ELAN and GhostConv-based YOLO network (EGCY-Net) is proposed with a Conv-GhostConv Stack (CGStack) module and an ELAN-GhostConv Network (EGCNet). To enhance inter-channel relationships, the CGStack module captures complex patterns and information in the image by using ghost convolution to increase the model inference speed while retaining the ability to capture spatial features. EGCNet is designed and constructed based on ELAN and the CGStack module to capture and utilize hierarchical features efficiently in layer aggregation. Additionally, the proposed methodology involves the creation of a dataset comprising images of boxes taken in warehouse settings. The proposed system is realized on the NVIDIA Jetson Nano platform, using an Arducam IMX477 camera. To evaluate the proposed model, we conducted experiments with our own dataset and compared the results with some state-of-the-art (SOTA) models. The proposed network achieved the highest detection accuracy with the fewest parameters compared to other SOTA models.

**Keywords:** object detection; AGV; package recognition



**Citation:** Firdiantika, I.M.; Lee, S.; Bhattacharyya, C.; Jang, Y.; Kim, S. EGCY-Net: An ELAN and GhostConv-Based YOLO Network for Stacked Packages in Logistic Systems. *Appl. Sci.* **2024**, *14*, 2763. <https://doi.org/10.3390/app14072763>

Academic Editors: Alexander N. Pisarchik and Xi Yang

Received: 7 February 2024

Revised: 7 March 2024

Accepted: 17 March 2024

Published: 26 March 2024



**Copyright:** © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

In supply chain management, the movement of materials and goods through several network points demands manual checks at each location. Inspections are conducted to ensure that the items being moved are correctly identified and the instructions are checked for completeness. In conventional manufacturing, stacked package recognition/movement is still a task performed by people. Owing to the time-consuming and complex factory environment, manual operations are no longer sufficient to meet production needs [1,2]. Automation is increasingly used across various scenarios, driven by the rapid expansion of the modern manufacturing industry. According to current trends in manufacturing and logistics, the fastest, most efficient, and least expensive system prevails in terms of logistics productivity. Since automated guided vehicles (AGVs) can operate autonomously and move things back and forth during manufacturing, they can be employed in factories to improve efficiency and quality. AGV systems function autonomously, requiring no human involvement, such as drivers. Software and sensor-based guidance systems operate together to manage their logistics. The usefulness of AGVs is that they can stimulate production by efficiently using inventory space and accelerating the logistics through

automation. Additionally, because work is conducted along a prearranged route, the chance of an accident occurring can be significantly decreased [2,3].

Containers are used to move material between locations during manufacturing. In logistics, AGVs will move containers to a specific location based on the instructions. Since many types of containers need to be recognized by an AGV in a factory, the necessity for precision, stability, and adaptability in AGV systems is important. Thus, vision-based technology is an ideal solution as deep learning and computer vision technologies continue to improve [4]. To enable vision-based technology, we provide an automated system for container recognition. This approach facilitates the sorting of containers, allowing them to be classified and counted more accurately and effectively throughout the manufacturing process.

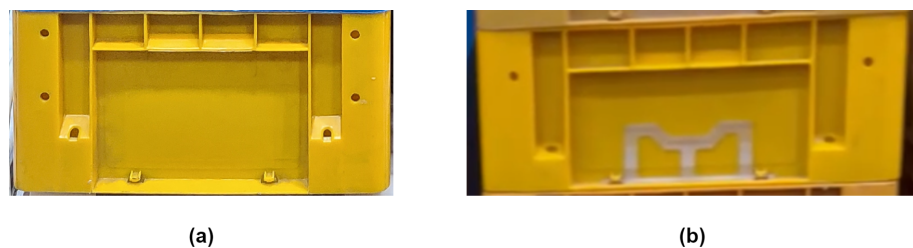
RFID technology, along with integrated IT platforms, is used in logistics organizations for warehousing needs. RFID utilizes electromagnetic fields to automatically recognize and monitor tags attached to items. RFID tags, especially passive ones that do not require batteries, are highly recognized, robust, and energy-efficient, guaranteeing long-term reliability and cost-effectiveness. RFID technology has advantages in inventory management, while vision-based techniques provide more promising benefits. RFID technology offers detailed data linked to each tag, including color, measurements, and price, which is advantageous for improving inventory control. Nevertheless, this technology has limitations regarding its flexibility and adaptability. RFID systems depend on certain tags requiring appropriate infrastructure and have difficulty adjusting to environmental variations or different object categories. RFID technology may have lower accuracy in real-time object detection, which could impact the speed of responding to inventory changes. Vision-based techniques provide flexibility, adaptability, and high precision in object detection. Images, especially those in color, provide extensive information that surpasses that of other sensors. In addition to providing distance information, they also recognize an object's color, texture, and depth. Vision systems may adjust to different environments and object categories without relying on specific tags, providing real-time and detailed information about inventory items. Additionally, vision-based technologies enable immediate responses to changes in the inventory status, making them superior options for modern logistical operations [1,5–7].

Automated package recognition systems have been developed by many companies. For example, a system developed by Logivations uses an automated method to identify products and their quantities. Algorithms derived from computer vision and AI are employed to decode labels, barcodes, and QR codes, read any text, and identify objects [8]. Regarding the recognition of boxes, numerous academic research groups have conducted studies that achieved significant advancements in the identification of packing structures [1,4,9–13]. A multi-step image processing pipeline was suggested by Dörr et al. for automated packaging structure recognition [1]. This pipeline begins with the localization and identification of logistics transport units, followed by the convolutional neural network (CNN)-based and image processing techniques to segment visible side faces. Packaging units are consolidated and classified to produce a single RGB image that shows the structure of each unit. In [4], Dörr and colleagues suggested using a deep learning instance segmentation model to detect and segment transport units in an input image, with the aim of recognizing and counting packages. This is followed by intra-unit segmentation to identify base pallets, transport unit sides, and package units. Information consolidation entails the allocation of packaging units to certain sides of transport units, refining the segmentation, and calculating the number of packages based on adjusted sizes. This process leads to the precise identification of packing structures inside transport units. Vasileva and Ivanovski [9] proposed a hybrid approach that aims to perform the real-time instance segmentation of packages using 2D distance maps by integrating both deep learning and digital signal processing techniques. Li et al. [10] suggested an approach for intelligent de-palletizing systems, which uses computer vision to recognize containers. The approach uses geometric-based structural elements, non-maximum suppression, better mean-shift clustering, and line merging. Naumann et al. [11] suggested a method that



involves reconstructing the shapes of undamaged and damaged parcels using RGB photos. It utilizes the CubeRefine R-CNN architecture, which combines 3D bounding box estimates with iterative mesh refinement. This approach allows detecting mistreatment and enables tampering detection. The study in [12] discusses automating the unloading of containers and suggests a technique that combines a Mask-RCNN algorithm, which is based on deep learning, with 3D depth measurements to detect targets. In [13], TetraPackNet (a unique model) is presented for segmenting objects by four vertices rather than bounding boxes or pixel masks. The model is used for recognizing the structure of logistics packaging.

Considering the related work above, there is no work similar to our research. Our research is important and difficult in that it considers the task of identifying specific types of boxes. These boxes have nearly identical shapes and are hard to separate into different layers stacked within them. For instance, in Figure 1, we can see that box 13 and box 04 have similar shapes, leading to the misclassification of box types by a deep learning model. This paper proposes a technique that utilizes vision-based package recognition. Our task is two-fold: (1) identify an empty location within the usable storage area to load the box; and (2) identify the category and the number of boxes.



**Figure 1.** The subfigures (a,b) depict Box class 13 and 04, respectively. These classes exhibit similar shapes and dimensions in terms of length and width, while differing notably in height.

Although current detection approaches employing deep learning techniques provide high precision, they encounter many constraints, such as an excessive number of parameters, poor system runtime rates, and the occurrence of false positives. Hence, this study suggests an ELAN and GhostConv-based YOLO network (EGCY-Net) for stacked package detection that applies YOLOv7 [14] and combines ELAN design techniques with GhostConv and other operations. The following summarizes the main contributions of this study.

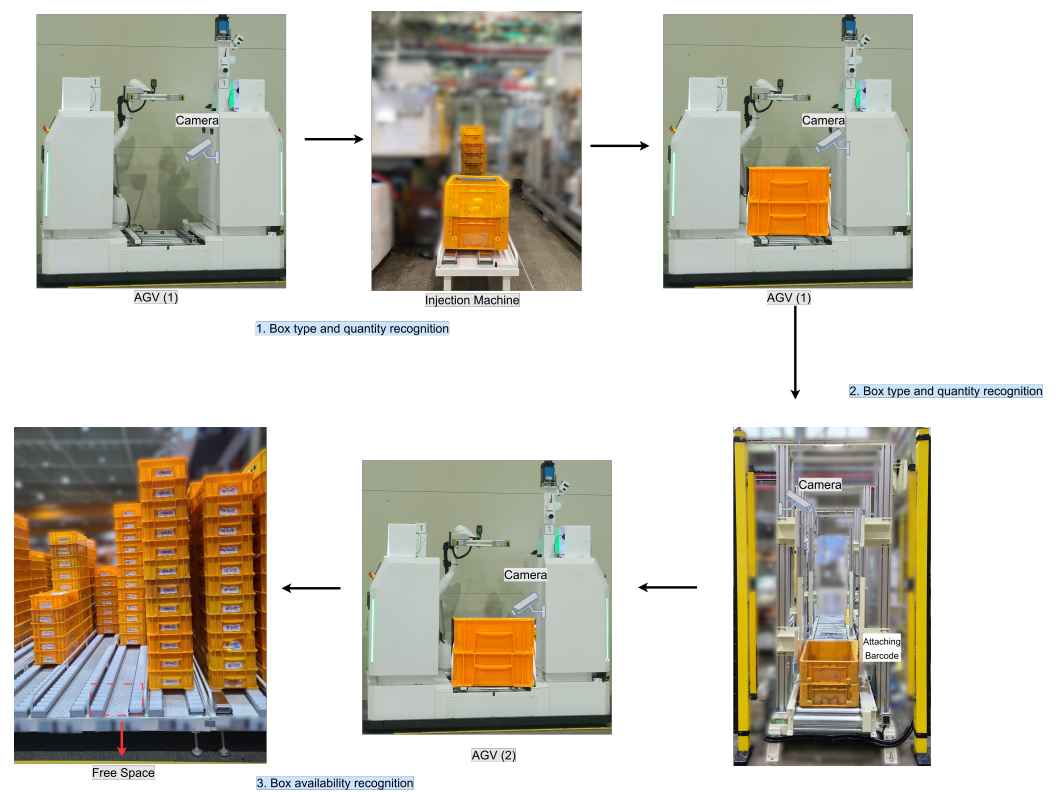
- To the best of our knowledge, we are the first to apply one-stage detection to tackle the task of identifying types of boxes and counting the number of boxes.
- To tackle the problem of capturing complex patterns and information in the data, we propose the implementation of a Conv-GhostConv Stack (CGStack) block that consists of a  $1 \times 1$  convolution, ghost convolution, and another  $1 \times 1$  convolution.
- The framework of the ELAN-based ghost convolution network (EGCNet) is proposed. It includes three convolution layers, two CGStack blocks, and concatenation with ELAN-based design strategies. This module helps the model to aggregate information across many layers.
- We built a dataset to compare the suggested approaches (in terms of classification accuracy) with several state-of-the-art (SOTA) methods, to demonstrate the practicality and applicability of the proposed method.
- By comparing our method with other SOTA methods, EGCY-Net improves the inference speed while balancing detection accuracy. It provides significant advantages to implementing this method in an embedded system.

The rest of this paper is structured as follows. Section 2 details the EGCY-Net architecture. Section 3 explains the results from comparison and ablation experiments. The discussion, limitations, and future works are presented in Sections 4 and 5, respectively. Our conclusions are presented in Section 6.

## 2. Materials and Methods

At the beginning of the process, the automated guided vehicle drives to the injection machine in order to retrieve boxes that contain raw materials. In the next step, it identifies the type of box and counts the total number of boxes. After that, the AGV transports the boxes to the barcode attachment area. As soon as the procedure is over, since the boxes might be replaced by human workers, the AGV once again recognizes the type of box and counts them. Following the completion of this procedure, a second AGV will relocate the boxes from the barcode's attachment area to an empty location where they are to be kept. Figure 2 presents a series of photos of the workflow. These images were captured at the Pyeong Hwa Automotive factory, Daegu, Republic of Korea.

There are two distinct activities that need to be completed in order to effectively ease the movement of boxes from various locations. The first thing that needs to be conducted is to identify the different types of boxes and determine the quantity in each category. The second task is to locate empty places in the storage sections. From a technical standpoint, our study makes use of the current developments in the field of image processing, including object classification and object identification. We applied MobileNet, a state-of-the-art neural network for object classification [15], to identify empty spaces in areas where boxes are stored. In addition, EGCY-Net is proposed as a method to identify various types of boxes.



**Figure 2.** Stacked package recognition workflow.

### 2.1. Box Availability Recognition

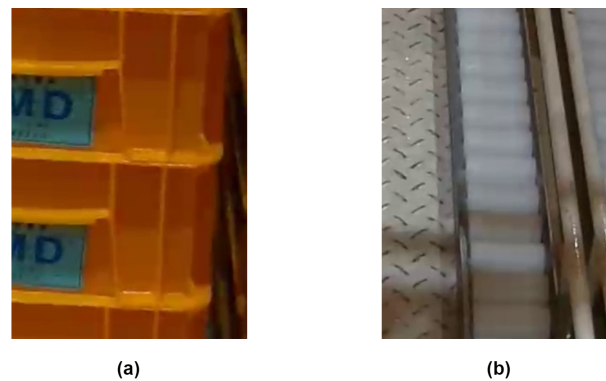
#### 2.1.1. The Dataset

The dataset for box availability recognition has two classes: box and no box. Since the dataset will be used for classification, an image that only contains one class is needed. Initially, a video was taken in the Pyeong Hwa Automotive Factory, Daegu, Republic of Korea. However, in a real environment, there are empty spaces and box spaces in the video. Therefore, the desired region of interest (RoI) should be applied, and the video should be cropped to the desired RoI. Figure 3 shows the image captured from the video before and after selecting the RoI.



**Figure 3.** Selecting the RoI.

After the RoI is selected, frames are extracted from the video, resulting in 1200 images for the ‘box’ category and 1200 images for the ‘no box’ category. All images are in JPG file format. Figure 4 shows a sample from the box availability recognition dataset.



**Figure 4.** Example of binary classification: (a) box, and (b) no box.

### 2.1.2. The Network Architecture

MobileNet [15] is a family of lightweight convolutional neural network architectures designed for efficient deployment in mobile and embedded devices. The MobileNet architecture is characterized by its innovative use of depth-wise separable convolutions, which efficiently reduces the computational burden and model size. The network comprises multiple convolutional blocks, each featuring a depth-wise convolution step that processes input channels separately, significantly lowering the computational cost. This is followed by point-wise convolution that combines the features learned from depth-wise convolution, enhancing representational power. Batch normalization and activation functions are applied after each convolutional operation to normalize the output and introduce nonlinearity. Toward the end of the network, global average pooling is often applied to generate a fixed-size tensor. All layers are followed by batch norm and ReLU nonlinearity, except for the final fully connected layer, which has no nonlinearity, and feeds into a softmax layer for classification.

## 2.2. Box Type and Quantity Recognition

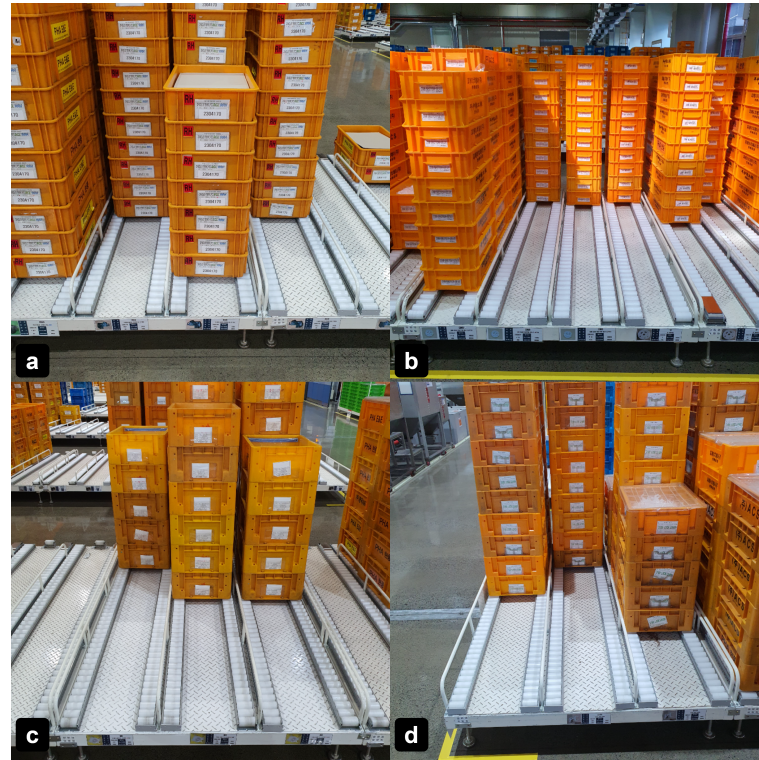
### 2.2.1. The Dataset

This research selected an object detection method based on deep learning. The success of deep learning in computer vision relies significantly on the availability of large datasets to avoid overfitting, especially when the neural network parameters are extensive. The more sufficient the dataset, the more effective features the model can extract, and the better the model fit. The dataset collected was captured in the Pyeong Hwa Automotive Factory. The sample images in the experiment were mainly from images and videos of boxes taken with mobile phones. All images are 1000 to 3000 pixels wide and 2000 to 5000 pixels high in a JPG file format. The dataset consists of boxes classified into four types: 04, 13, A, and Y.



Several frames were extracted from the videos. Since many of the frames were identical, eliminating some of them was necessary. The final dataset contained 1258 images; Figure 5 shows examples.

The training and inference speed of the neural network will be reduced if the image resolution is too high. Since there are many types of boxes and backgrounds in one image, we cropped images to one stack, as seen in Figure 6. After cropping the image, more than 400 images are acquired for each class.



**Figure 5.** Example images: (a) Class Y, (b) Class A, (c) Class 13, and (d) Class 04.



**Figure 6.** Cropping an image for the final dataset.

Labeling datasets is one of the most important steps in object detection. To make a deep learning model learn properly, we need to neatly and correctly label each object in the images. After completing the image cropping, the four box classes were manually annotated using MakeSenseAI 1.10.0-alpha [16], as seen in Figure 7. During the annotation process, the rectangular tool was selected to annotate each stack of boxes. The marked

labels were saved as plain text files. Each image corresponds to a label file. Every line in the label file has five numbers that represent an instance. The five numbers represent the class, the center of the bounding box (x), the center of the bounding box (y), the normalized width, and the normalized height. Box information can be seen in Table 1 and the image numbers in the final dataset are shown in Table 2.

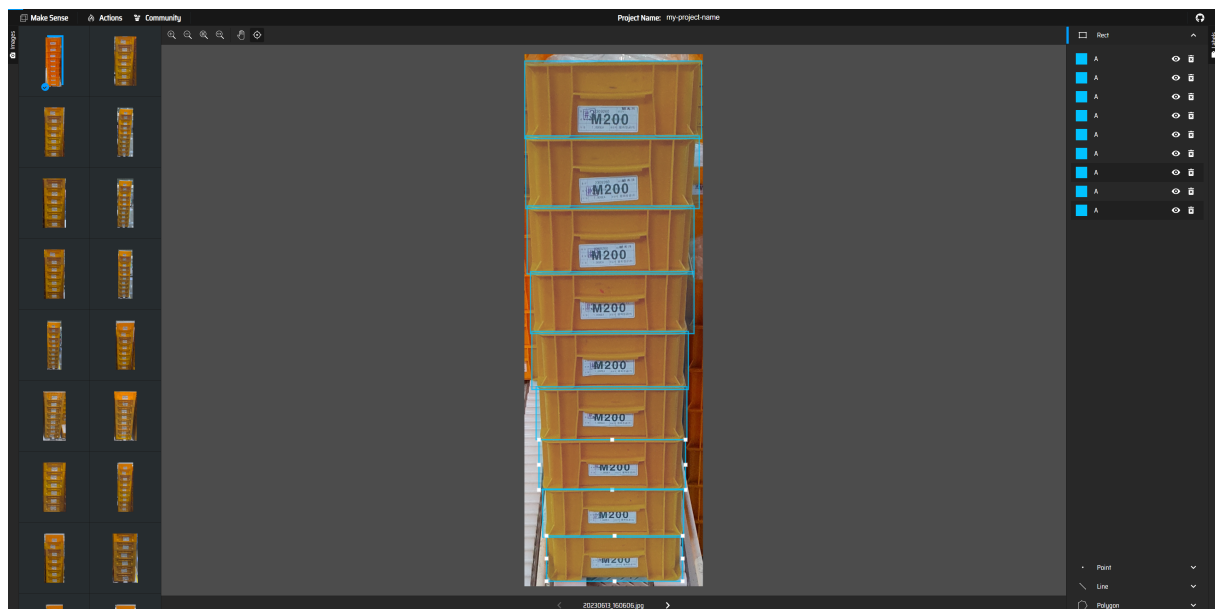


Figure 7. Dataset labeling.

Table 1. Box type information.

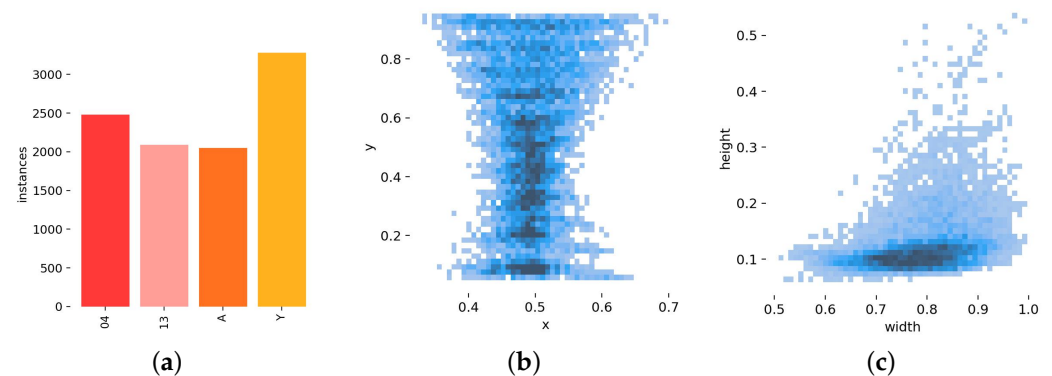
| Name of Box | Size (Length $\times$ Width $\times$ Height)<br>[mm] | Picture                                                                               |
|-------------|------------------------------------------------------|---------------------------------------------------------------------------------------|
| A           | 380 $\times$ 240 $\times$ 105                        |  |
| Y           | 590 $\times$ 585 $\times$ 150                        |  |
| 04          | 480 $\times$ 380 $\times$ 150                        |  |
| 13          | 480 $\times$ 380 $\times$ 200                        |  |

Table 2. The number of images in the dataset.

| Class | Train | Val | Test |
|-------|-------|-----|------|
| 04    | 420   | 80  | 12   |
| 13    | 420   | 80  | 34   |
| A     | 420   | 80  | 40   |
| Y     | 420   | 80  | 22   |



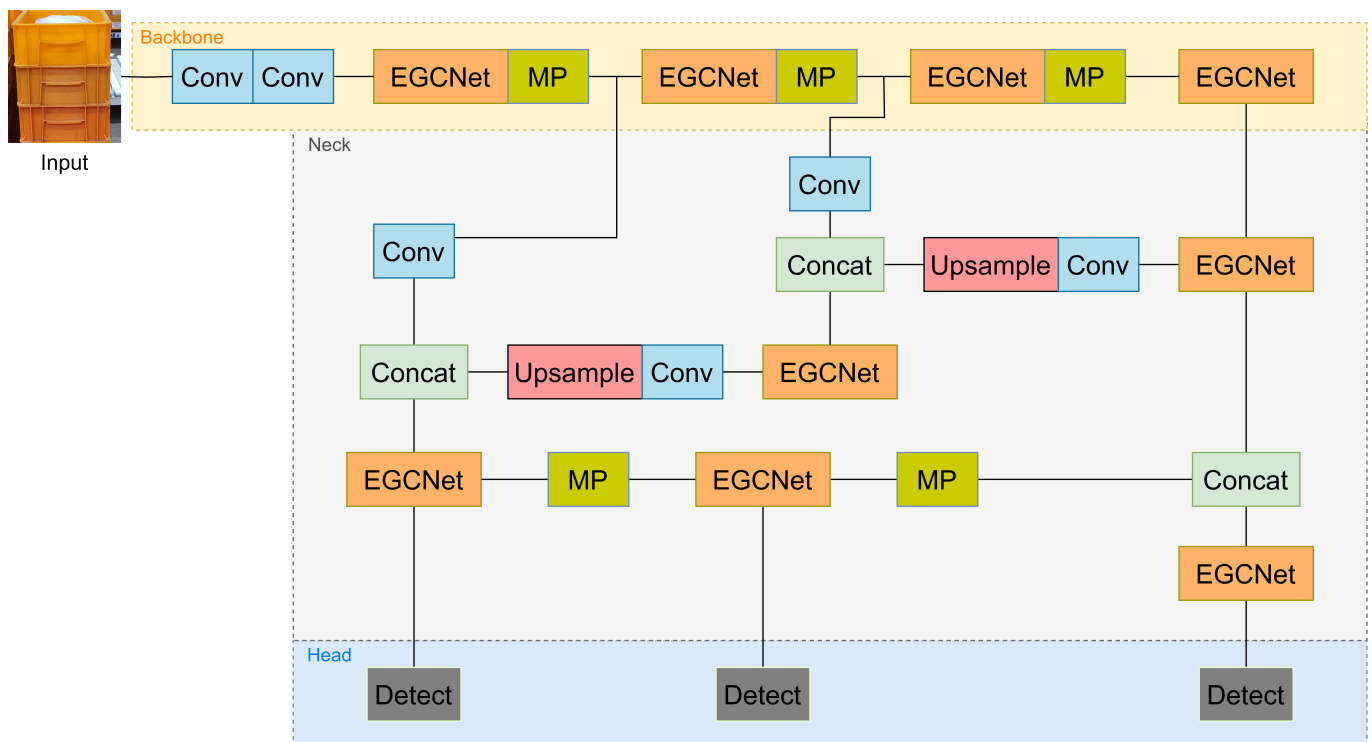
The numbers and distributions of objects in the training set are shown in Figure 8. Figure 8a shows the object names and corresponding amounts, indicating that the dataset encompasses an adequate number of instances for each box type. Figure 8b shows the distribution of tags. The x-axis is the ratio of the label center to the image width, and the y-axis is the ratio of the label center to the image height. As seen in the figure, the data are widely distributed and concentrated in the middle of the image. Figure 8c shows the sizes of the objects. The x-axis is the ratio of the label width to the image width, and the y-axis is the ratio of the label height to the image height.



**Figure 8.** Labels and label distributions: (a) number of labels; (b) label locations; and (c) sizes.

### 2.2.2. The EGCY-Net Architecture

The network structure for EGCY-Net proposed in this paper is shown in Figure 9; it is mainly composed of the backbone, neck, and head. This network was constructed based on YOLOv7 code, but most of the EGCY-Net structure is different from YOLOv7 [14].



**Figure 9.** Structure of the EGCY-Net.

The EGCY-Net backbone contains Conv, EGCNet, and MP layers, which are used for feature extraction to ensure completeness. The EGCY-Net neck includes Conv, Upsample, MP, concatenation, and EGCNet with multi-scale feature fusion mode to integrate low-level

spatial information and high-level semantic features to preserve more details and improve detection accuracy. This combines feature information at different scales. It continues to extract features from valid feature layers already obtained, upsampling them for feature fusion, and downsampling the features again for feature fusion. The head selects IDetect Head with three target sizes: large, medium, and small. The head is used as a classifier and regressor, and through the backbone and neck, three enhanced adequate feature layers are obtained, which are input for the head in decoupling feature information to output the position, confidence, and target type.

### 2.2.3. The GhostConv Module

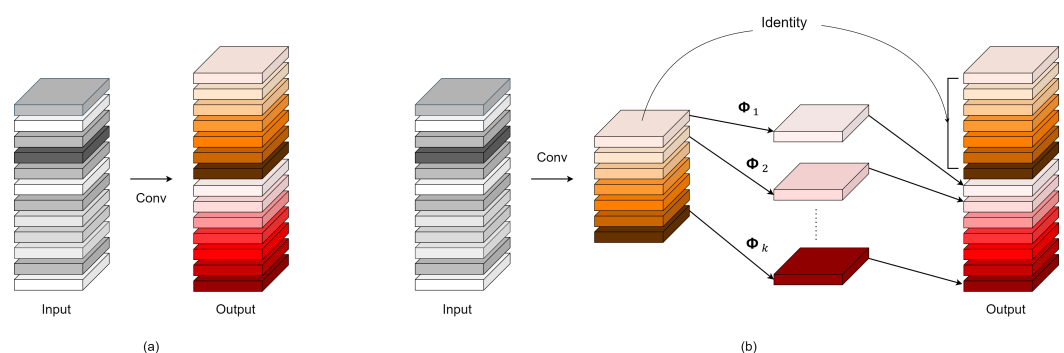
Neural networks usually contain extensive numbers of parameters, particularly in fully connected layers. The advancement of convolutional neural networks has made it possible to use filters that drastically cut down on the number of parameters. In order to create a network suited to performing detection tasks, a large model is generally required, which in turn demands a lot of feature maps, each of which typically contains hundreds of channels. The model is bloated because of the number of layers, even in SOTA models. By reducing the number of parameters in a neural network, we can make it easier to deploy on devices. Illustrations of the convolution layer and GhostConv are presented in Figure 10.

The input feature map is represented by  $X \in R^{c \times h \times w}$ , where  $c$  denotes the number of channels in the input feature map, and  $h$  and  $w$  denote the feature map's height and width, respectively. The standard convolution method is presented in Equation (1), as follows:

$$Y = X * f + b. \quad (1)$$

In that equation,  $Y \in R^{n \times h' \times w'}$  represents an output feature map with  $n$  channels, where  $h'$  and  $w'$  represent the height and width, respectively, of the output feature map. The convolution filters are denoted by  $f \in R^{c \times k \times k \times n}$ , and  $*$  indicates the convolution operation. The size of the convolution kernel is  $k * k$ , and  $b$  represents the bias term. The regular convolution computation, excluding the bias term, is approximately equal to  $n \times h' \times w' \times c \times k \times k$ . In a shallow layer of the network, the values of  $h'$  and  $w'$  are more expansive, whereas in a deeper layer, the values of  $n$  and  $c$  are larger. The ghost convolution [17] method was introduced, which comprises two components: a regular convolution kernel that produces a limited number of feature maps and the creation of additional feature maps in a lightweight linear transform layer. This can be mathematically represented as follows:

$$Y' = X * f' + b. \quad (2)$$



**Figure 10.** (a) The convolution layer and (b) the ghost convolution layer.

Equation (2) represents a standard convolutional layer that produces only a small number of feature maps. Here,  $Y' \in R^{h' \times w' \times m}$  represents the resulting feature output, and  $f' \in R^{c \times k \times k \times m}$  represents the size of the convolutional kernel used. The output feature map has fewer channels compared to the standard convolutional layer, specifically  $m < n$ .

$$y_{ij} = \phi_{i,j}(y'_i). \quad (3)$$

The linear transformation layer—responsible for producing redundant feature maps—is represented by Equation (3), where  $y_i$  represents the  $m$  feature maps of  $Y'$ . In order to generate  $s$  feature maps, a lightweight linear transformation operation is applied to each feature map in  $Y'$ . When the  $d \times d$  convolution is employed as the linear transform, the final linear transform is mandatorily specified as a constant transform. Consequently, linear transformations of the  $m \times (s - 1)$  feature maps yield  $m$  feature maps. Using ghost convolution, the total computation is  $(s - 1) \times m \times h' \times w' \times k \times k$ .

#### 2.2.4. The Conv-GhostConv Stack (CGStack) Module

A standard convolutional layer in a neural network is denoted with Conv. The layer has batch normalization (nn.BatchNorm2d) and the SiLU activation function to make the data less linear. Based on the Conv layer, GhostConv is proposed to reduce the parameters. The ghost convolution method splits the convolutional operation by dividing input channels into two groups and employing  $1 \times 1$  and  $3 \times 3$  convolutional kernels for feature extraction. The results are then combined in the concatenation layer.

The bottleneck structure [18] consists of three layers: a  $1 \times 1$  convolutional layer with fewer filters, the reduced-dimension representation processed by a  $3 \times 3$  convolutional layer, and another  $1 \times 1$  convolutional layer to restore dimensionality to the original size. In CGStack, the  $3 \times 3$  convolutional layer in the bottleneck structure is replaced with a ghost convolution, introducing a different type of operation into the architecture. Ghost convolution is a technique designed to reduce computational costs and the model size by approximating the output of a standard convolutional layer. Based on Conv and GhostConv [17], CGStack is proposed to enhance the flexibility and efficiency of convolutional layers in deep learning architectures, which can be seen in Figure 11. This convolution layer in the module helps the model enhance the inter-channel relationship and the GhostConv layer reduces the number of parameters while preserving the ability to capture spatial features. The module allows the network to efficiently learn both low-level and high-level features: edges, colors, textures, shapes, and the entire object in the image.

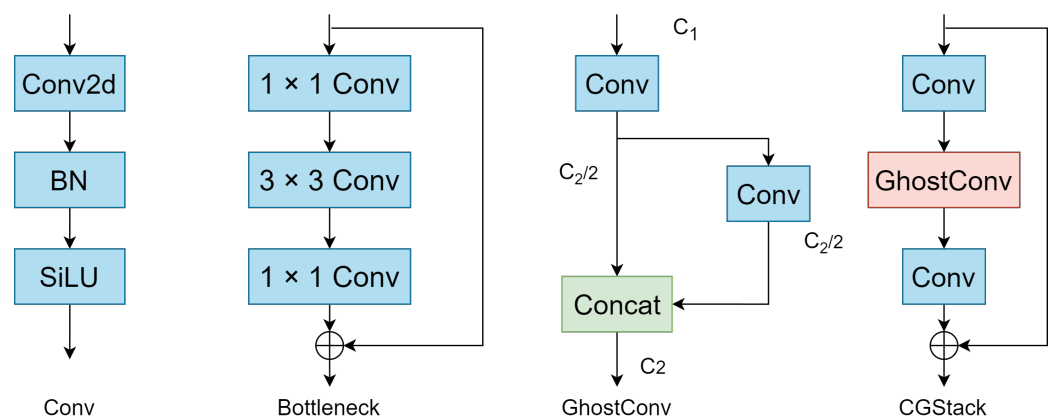
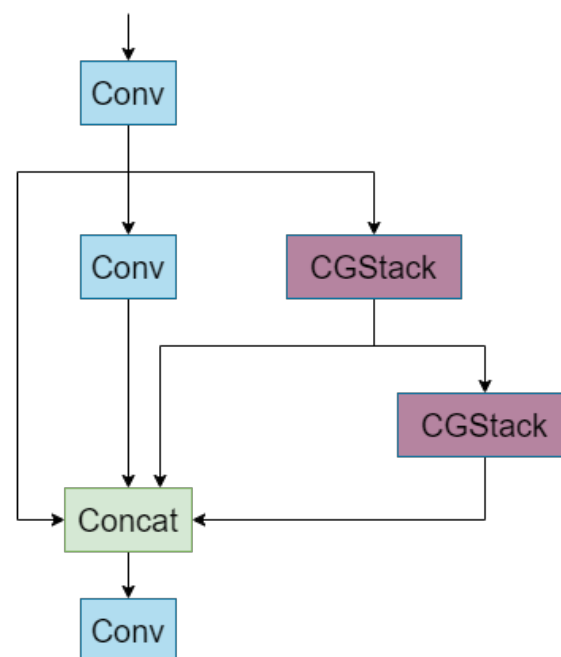


Figure 11. The CGStack structure.

### 2.2.5. The ELAN-GhostConv Network

Model scaling refers to the process of adjusting the size of a pre-existing model to ensure compatibility with various computing devices, whether it be increasing or decreasing its dimensions. The model scaling method commonly employs various scaling factors, such as resolution (input image size), depth (layer number), width (channel number), and stage (feature pyramid number), to achieve an optimal compromise between the network parameters, computation, inference speed, and accuracy [19–21]. During the process of model scaling, a problem occurs when the network reaches a specific depth. If we continue to add computing blocks, the accuracy will gradually diminish. Furthermore, once the network reaches a certain depth, its convergence starts to decline, leading to overall accuracy that is inferior to that of shallow networks [22].

The main purpose of designing ELAN [22] is to address the issue of gradual degradation in deep model convergence during model scaling. The ELAN structure helps to analyze the shortest and longest gradient pathways throughout each layer in the network, enabling the creation of a layer aggregation architecture that ensures the effective propagation of gradients. According to the previous explanation, we put forward a network that has a comparable ELAN structure to enrich features from aggregation layers. To achieve this, we utilize Conv, the CGStack module, and concatenation to construct EGCNet. The proposed EGCNet structure is depicted in Figure 12.



**Figure 12.** The EGCNet module structure.

### 2.3. Counting the Boxes

Once the box class has been identified, the next task is to ascertain the quantity of boxes. This approach is specifically designed to calculate the number of boxes within cropped frames corresponding to the region of interest (RoI). The box is considered valid if its dimensions surpass the minimum height and width specifications and if the score exceeds 50. Subsequently, a class will be generated for each box detected, and the number of boxes will be calculated based on the output of the class. Algorithm 1 is a pseudocode for the proposed method of counting boxes.



**Algorithm 1:** Box counting pseudocode

---

```

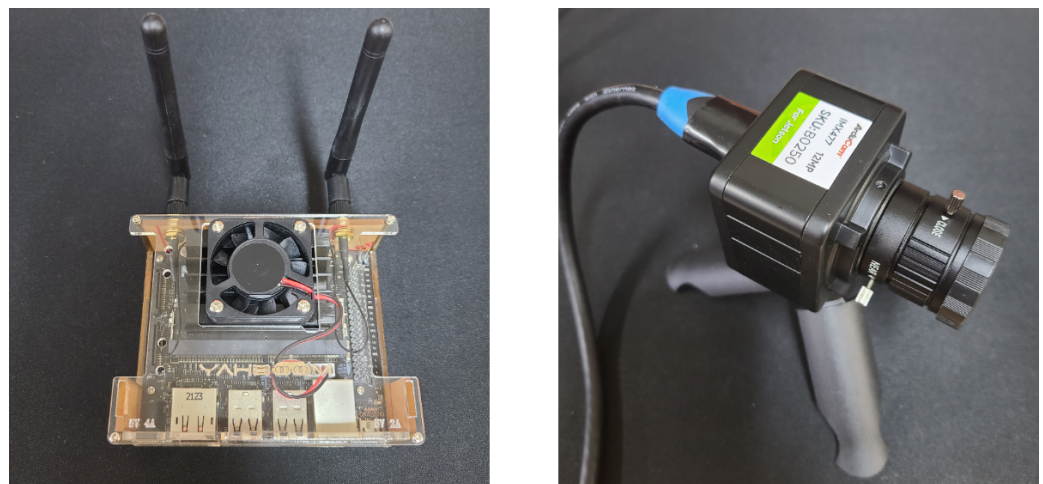
Input: cropped_frames
Output: output
boxes_list = [] /* to store detected boxes */
classes_count = {} /* to store the count of each class */
for each box in the RoI do
    if box.size  $\geq$  min_height_size  $\times$  min_width_size and box.threshold > 50 then
        detected_class = detect_class(box);
        boxes_list.append((box, detected_class));
    end
end
Count the number of boxes based on the detected classes:
for (each box, detected_class) in boxes_list do
    if detected_class not in classes_count then
        classes_count[detected_class] = 1;
    end
    else
        classes_count[detected_class] += 1;
    end
end

```

---

**2.4. Implementation**

Deep learning algorithms have significant computing costs throughout the training process, particularly when dealing with large-scale datasets. This research utilized NVIDIA's Jetson Nano and the Arducam IMX477 camera from ArduCAM (Figure 13). The Jetson Nano is a compact high-performance microprocessor specifically engineered to operate as an embedded device running artificial intelligence applications. The Jetson platform is equipped with a 64-bit ARM quad-core CPU, a 128-core integrated NVIDIA GPU, and 4 GB of LPDDR4 memory. The Jetson Nano is useless without an operating system. Therefore, from a micro SD card, we installed JetPack 4.6.1, which uses L4T version R32.7.1.



**Figure 13.** Left, the Jetson Nano; right, the Arducam IMX477 camera.

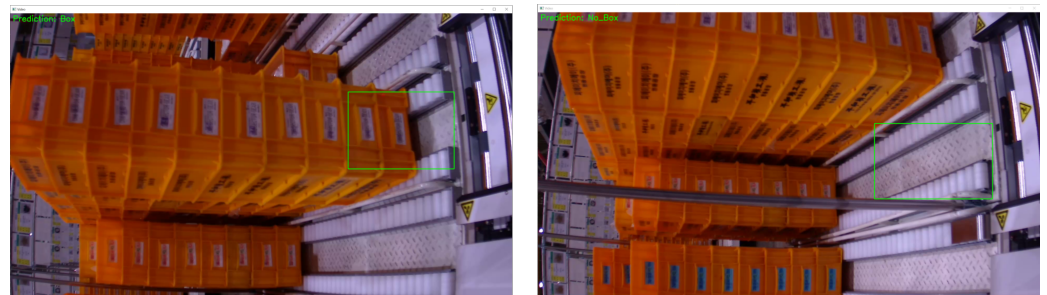
The Arducam CSI-USB UVC Camera Adapter Board was utilized for the 12.3 MP IMX477 camera. A dual-core NVIDIA GeForce RTX 4090 GPU was employed in the experiments during training to facilitate the training and assessment of the object detector model. The operating system used was Linux, supplemented with NVIDIA CUDA 11.8 and cuDNN 8.1 for GPU acceleration.

Once the MobileNet and object detector models are trained, they must be converted from PyTorch (.pt) to ONNX format. The ONNX format can be beneficial when implementing the model on devices like the Jetson Nano because it is a more widely supported format for inference across various frameworks and hardware platforms. The interoperability of the ONNX format enables its utilization across a multitude of deep learning frameworks and hardware accelerators. By converting the model to ONNX format, the probability that it can be executed on the Jetson Nano increases.

### 3. Results

#### 3.1. Box Recognition

The input dimensions of the images were 128, 128, and 3, and training was conducted over 15 epochs. With a batch size of 8, 1200 images were utilized per class for an overall total of 2400 images. Following the model's training, a demonstration video incorporated both examples, as shown in Figure 14. The predicted class will appear as illustrated. If the box is detected within the video, the text Box is displayed; otherwise, No Box appears. The image indicates that the classes were accurately detected using the constant ROI.



**Figure 14.** Results of box recognition. The green box highlights the Region of Interest (RoI) where the model concentrates its detection efforts on objects within this specified area.

#### 3.2. Box Type and Quantity

##### 3.2.1. Evaluation Metrics

To verify the efficacy and performance of the network model for stacked package recognition, we used precision, recall, mean average precision (mAP) @.5, and @.5:.95, alongside the parameters and model size in YOLOv7, as the evaluation indicators for the EGCY-Net model and the other baseline detection models. Higher values of precision, recall, mAP@.5, and mAP@.5:.95 with fewer parameters and a smaller model size are ideal in stacked package detection.

The formula for calculating mAP is as follows:

$$mAP = \frac{1}{n} \sum_{i=1}^n AP_i \quad (4)$$

The term “mAP@.5” indicates the mean average precision is calculated with a threshold greater than 0.5, while mAP@.5:.95 indicates the mean average precision computed at various intersection over union (IoU) thresholds ranging from 0.5 to 0.95, with increments of 0.05. Precision is a quantitative measure that evaluates the accuracy of an object identification algorithm by calculating the ratio of accurately predicted positive instances to all instances predicted as positive. Recall, sometimes referred to as sensitivity or the true positive rate, quantifies the ability of an object identification system to identify positive cases. It computes the ratio of accurately predicted positive instances to the total number of actual positive instances. The term parameters refers to measurements that indicate the amount of memory used by a model. Furthermore, the model size gives the amount of memory the model occupies (in megabytes).

### 3.2.2. Experiment Results

To evaluate whether the proposed method has an advantage over state-of-the-art studies, several remarkable CNNs for object detection were chosen for comparison, namely, YOLOv3 [23], YOLOv5l [24], YOLOR [25], and YOLOv7 [14]. All the comparative methods were reproduced using the default settings given by the authors, and all models were trained for 20 epochs only. The overall results are in Table 3. After all models were trained for 20 epochs, we chose the best model to be trained with more epochs for implementation in Jetson Nano.

The following conclusions can be drawn from Table 3:

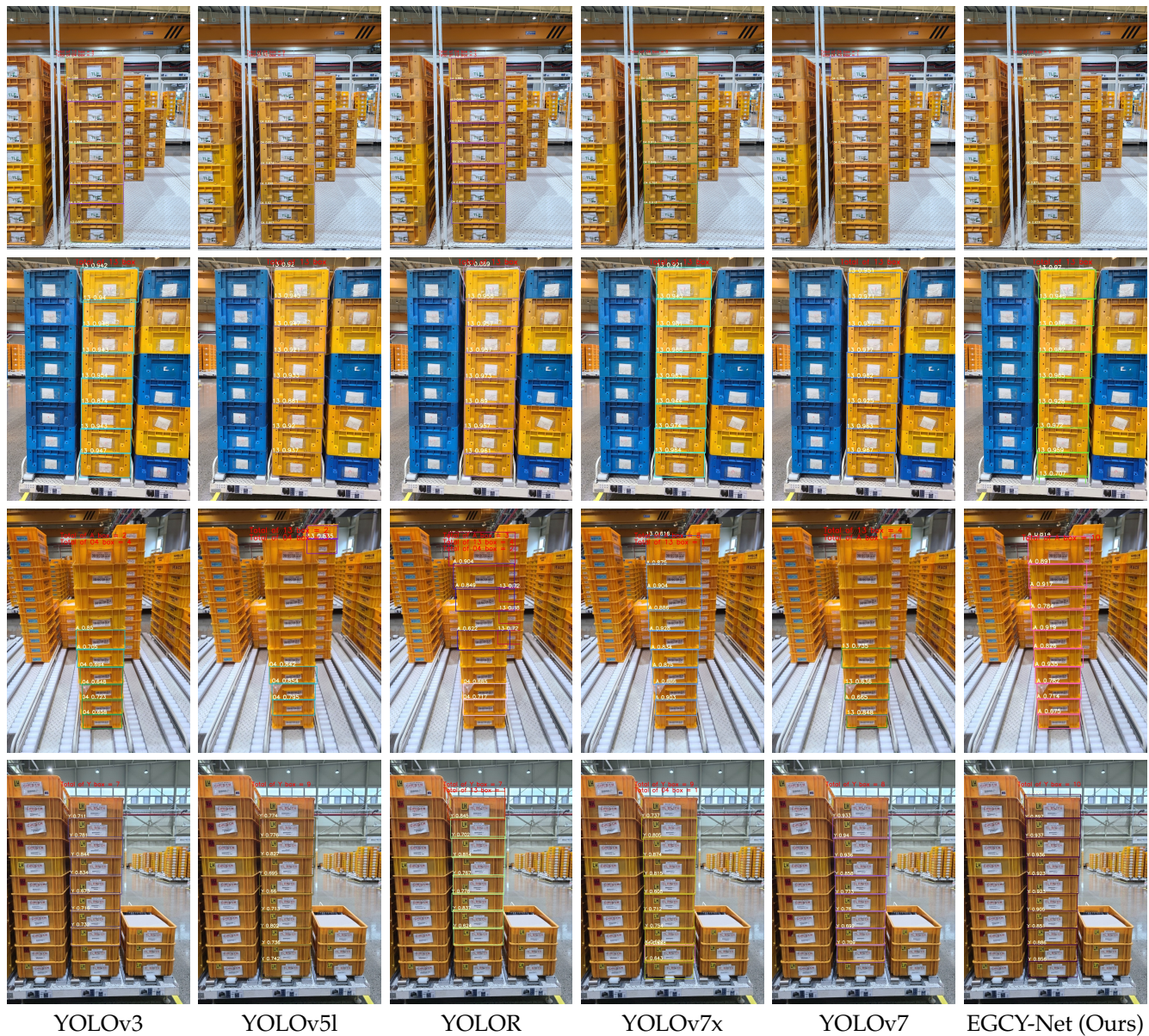
1. The parameter count of the EGCY-Net is approximately 30 M, while the size of the ONNX model is 111 MB. The result was the lowest compared to the other SOTA models in the experiment. In addition, the EGCY-Net testing time was 5.96 seconds, making it the fastest compared to the others.
2. EGCY-Net achieved mAP scores of 87.1%, 86.5%, 87.3%, 86.6%, and 88.5% for all, 04, 13, A, and Y classes, respectively, when evaluated at the IoU threshold range of 0.5 to 0.95.
3. The proposed network in YOLOv7x [14] achieved good results. Nevertheless, this approach had the highest number of parameters and the largest ONNX model size compared to the other algorithms.
4. YOLOv3 [23] had greater precision compared to YOLOv5l [24], YOLOv7 [14], and YOLOR [25], although it had more parameters and a larger model size. The YOLOv7 [14] also performed better when the size of the parameters was small. Additionally, YOLOv5l [24] had the third-fewest parameters, behind YOLOv7 [14] and EGCY-Net, but its performance was inferior to the other two models.

**Table 3.** Comparison results between EGCY-Net and SOTA models.

| Class                | Model         | YOLOv3 [23] | YOLOv5l [24] | YOLOR [25] | YOLOv7x [14] | YOLOv7 [14] | EGCY-Net (Ours) |
|----------------------|---------------|-------------|--------------|------------|--------------|-------------|-----------------|
| All                  | P (%)         | 99.4        | 98.9         | 99.3       | 99.5         | 99.6        | 99.6            |
|                      | R(%)          | 99          | 98.9         | 99.1       | 99.6         | 99.7        | 99.9            |
|                      | mAP@.5(%)     | 99.4        | 99.5         | 99.5       | 99.5         | 99.5        | 99.5            |
|                      | mAP@.5:.95(%) | 84.8        | 83.7         | 84         | 86.3         | 84.4        | <b>87.1</b>     |
| 04                   | P (%)         | 99.6        | 98.7         | 99         | 99.7         | 99.5        | 99.8            |
|                      | R(%)          | 99          | 98.8         | 98.7       | 99.6         | 99.2        | 99.5            |
|                      | mAP@.5(%)     | 99.5        | 99.5         | 99.5       | 99.6         | 99.5        | 99.5            |
|                      | mAP@.5:.95(%) | 84.6        | 83           | 83.6       | 85           | 82          | <b>86.5</b>     |
| 13                   | P (%)         | 99.3        | 99.1         | 98.8       | 99.3         | 99.3        | 99.1            |
|                      | R(%)          | 99.7        | 99.5         | 99.1       | 99.8         | 99.7        | 100             |
|                      | mAP@.5(%)     | 99.5        | 99.5         | 99.5       | 99.5         | 99.5        | 99.5            |
|                      | mAP@.5:.95(%) | 84.1        | 84.1         | 84.5       | 85.9         | 85.1        | <b>87.3</b>     |
| A                    | P (%)         | 99.3        | 99.5         | 99.8       | 99.7         | 99.8        | 99.8            |
|                      | R(%)          | 97.5        | 98.1         | 98.7       | 99.6         | 100         | 100             |
|                      | mAP@.5(%)     | 99.4        | 99.4         | 99.5       | 99.4         | 99.5        | 99.5            |
|                      | mAP@.5:.95(%) | 85.9        | 84.1         | 84.9       | <b>86.6</b>  | 84          | <b>86.6</b>     |
| Y                    | P (%)         | 99.4        | 98.4         | 99.7       | 99.2         | 99.7        | 99.7            |
|                      | R(%)          | 100         | 99.3         | 99.9       | 99.6         | 99.9        | 100             |
|                      | mAP@.5(%)     | 99.4        | 99.5         | 99.4       | 99.5         | 99.5        | 99.5            |
|                      | mAP@.5:.95(%) | 84.5        | 83.8         | 83.1       | 87.6         | 86.5        | <b>88.6</b>     |
| Parameter (M)        |               | 61.5        | 46.1         | 52.4       | 70.8         | 36.4        | <b>30</b>       |
| ONNX model Size (MB) |               | 234         | 176          | 200        | 270          | 139         | <b>111</b>      |
| Inference time (s)   |               | 13.6        | 17.32        | 14.829     | 12.995       | 12.744      | <b>5.96</b>     |

Bold score represents the best result achieved.

To demonstrate the efficacy of the proposed algorithm, images from the test subset were selected at random for comparison. The results from tests conducted using smartphone images are presented in Figure 15.



**Figure 15.** Comparisons of detection results using test images captured by a smartphone camera. **Left to right:** YOLOv3 [23], YOLOv5 [24], YOLOR [25], YOLOv7x [14], YOLOv7 [14], and our proposed method. The first, second, third, and fourth rows are classes 04, 13, A, and Y, respectively.

In addition, to evaluate the efficacy of the enhanced algorithm proposed in this paper, we chose a test image captured by the Arducam, which has a lens that is distinct from the smartphone camera. Test results with several images acquired by the Arducam are in Figure 16, which shows the network we propose on the right-hand side.

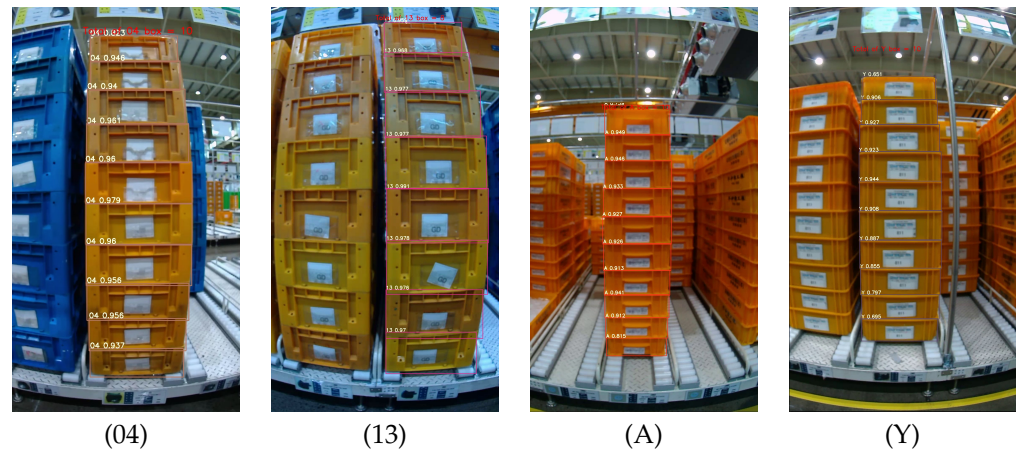




**Figure 16.** Detection results using test images captured with the Arducam. The first, second, third, and fourth rows are classes 04, 13, A, and Y, respectively.

From this result, EGCY-Net was trained over 300 epochs to increase its accuracy and minimize missed and false detections. After training for 300 epochs, EGCY-Net showed improved efficiency and a decrease in both missed and false detections. The visualization result can be seen in Figure 17.



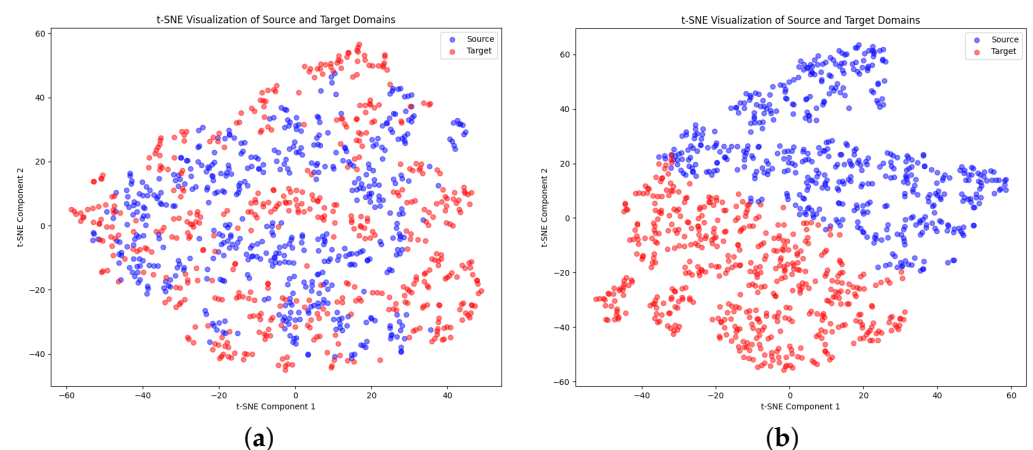


**Figure 17.** EGCY-Net results after training for 300 epochs. The first, second, third, and fourth columns are classes 04, 13, A, and Y, respectively.

### 3.2.3. Visualization of Feature Distribution

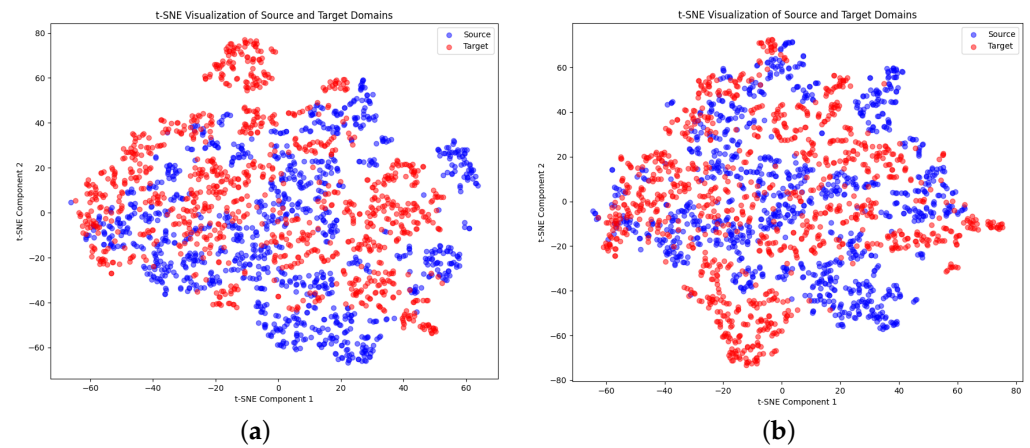
In this research, we used one domain adaptation scenario where the source domain consisted of images taken with a mobile phone camera, and the target domain comprised images captured by the Arducam. Figures 18 and 19 show the feature map visualization from EGCY-Net and YOLOv7: (a) features extracted from the same box class captured by different cameras, (b) features extracted from two different box classes captured by different cameras. The t-SNE statistical method [26] was used to reduce the dimensionality of the feature map output.

In Figures 18a and 19a, instances of the same category are close to each other, regardless of the camera used. This indicates that the features are robust across different devices, and the model generalizes well. In Figure 18b, instances of different object categories are easily distinguishable in different domains, meaning the model can capture variations between different types of objects. However, in Figure 19, the instances from different categories are not well-separated.

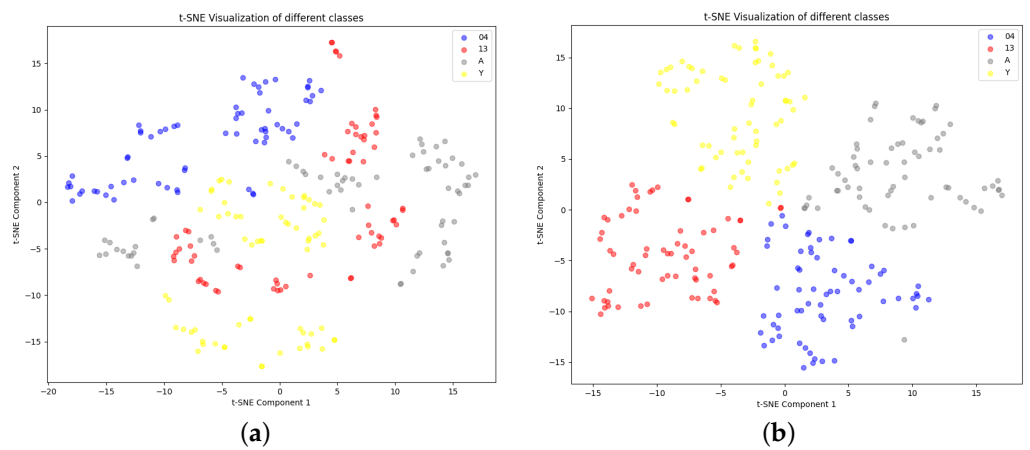


**Figure 18.** The EGCY-Net feature map visualization (before the detection layer).

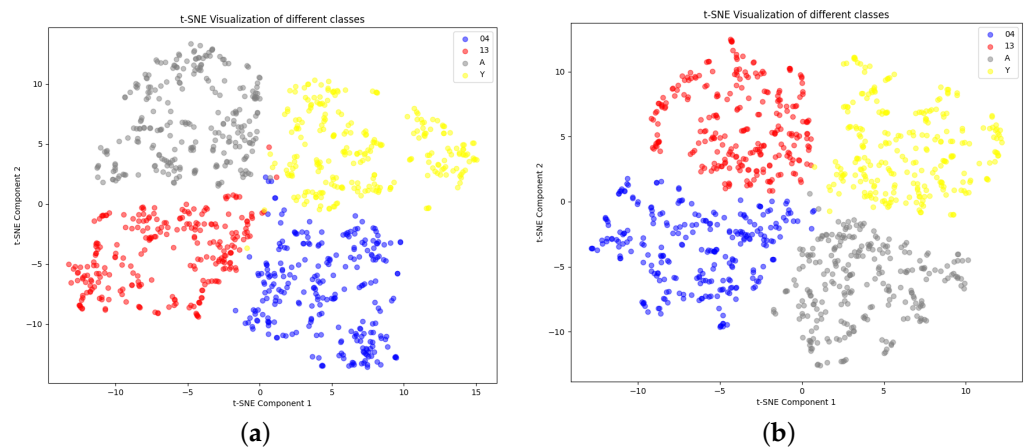
As shown in Figure 20, to analyze the effectiveness of the proposed EGCY-Net, features were extracted from Stage 4 of our proposed network and the baseline network (the CGStack layer and the Conv layer, respectively). The differences between ELAN in YOLOv7 and EGCNet in the proposed network are visualized in Figure 21. In YOLOv7 features, the model had difficulty distinguishing between certain classes, leading to different classes embedded close to each other. In EGCY-Net, instances belonging to different classes are well-separated (dots with different colors do not overlap).



**Figure 19.** The YOLOv7 feature map visualization (before the detection layer).



**Figure 20.** Visualization of feature maps from (a) YOLOv7 and (b) EGCY-Net.



**Figure 21.** Visualization of feature maps from (a) after the ELAN structure in YOLOv7 and (b) after the EGCNet structure in EGCY-Net.

#### 4. Discussion

It is challenging to detect objects with images captured by different camera lenses. In this experiment, we used images captured by a smartphone camera to train the model. However, in a real-time application, the Arducam camera was used. The differences in camera lenses may have an impact on the results. Therefore, we tested the model using a smartphone camera at first. YOLOv7 [14], YOLOv7x [14], YOLOR [25], YOLOv5l [24], and YOLOv3 [23] presented poor performance. For instance, detections by YOLOv3 [23],

YOLOv5l [24], YOLOR [25], and YOLOv7 [14] had false detection for Class 04 in the first row. Furthermore, YOLOv7x [14] accurately identified the classes but missed one box. All models in the second row performed admirably with the Class 13 box. YOLOv3 [23], YOLOv5l [24], YOLOR [25], YOLOv7x [14], and YOLOv7 [14] demonstrated inferior performance in detecting Class A in the third row. Class Y in the last row was correctly detected by YOLOv3 [23], YOLOv5l [24], and YOLOv7 [14], with only a few missed detections. Additionally, both YOLOR [25] and YOLOv7 [14] had false detections. However, EGCY-Net can recognize the box type correctly in smartphone images.

The proposed method was good at extracting features from the Arducam images of Classes 04 and 13, while YOLOv3, YOLOv5l, YOLOR, YOLOv7x, and YOLOv7 had missed and wrong detections. As seen in the third and fourth rows (Classes A and 13), all models had missed and false detections. However, among them, our proposed network and YOLOv7x showed better results for Class A (detecting nine boxes for Class A) but one false detection for Class 13. Meanwhile, the other models had false detection for three out of the four classes. For Class Y in the last row, all models had false detection, but YOLOv3 and YOLOv5 made better decisions for Class Y (nine correct detection). Based on the above, EGCY-Net had better results in detecting boxes in Arducam images and had the lowest ONNX size.

## 5. Limitation and Future Works

Although our study produced promising results, it has limitations related to the lack of dataset diversity, as our research mainly concentrated on a certain sort of object detection task. Expanding our dataset to include a wider range of scenarios and object kinds is a possible direction for future research.

This section discusses potential future strategies for improving object identification applications, specifically focusing on object identification in warehouses. We will further assess the efficacy of EGCY-Net for recognizing logistics packaging structures in our specific use case. Initially, we aimed to establish a connection between the Jetson Nano and AGV. We compared our model to various state-of-the-art models. We conducted experiments using several methodologies in various settings and conducted a comparative analysis for deeper insight. EGCY-Net is not simply limited to the use of recognizing package structures or logistics. We intend to confirm our positive findings by assessing EGCY-Net on further datasets with varying use cases.

## 6. Conclusions

The application requirement stipulates that stacked packages be precisely detected using only limited computing and storage resources, so the structure of each part in the YOLOv7 network was redesigned. Then, an EGCY-Net network was proposed for stacked packages in a logistics system. To reduce the computational costs of deep neural networks and to employ a model that is more suitable for the Jetson Nano, this paper presents EGCY-Net, an ELAN and GhostConv-based YOLO network used for building efficient neural architectures. By replacing the convolution kernel with a linear operation, these cheap operations save a lot of computing resources. Comparing mAP and the detection time, we conclude that the GhostNet module can reduce the size of the model while still retaining comparable performance. Moreover, the new Convolution-GhostConv Stack module and the ELAN-GhostConv network were proposed for EGCY-Net. EGCY-Net was tested on images captured by a mobile phone and by an Arducam IMX477 camera. The experimental results show that—compared with the other SOTA methods—the proposed EGCY-Net effectively improves object detection accuracy, reduces the parameters, and shortens the testing time.

**Author Contributions:** The contributions were distributed between the authors as follows: I.M.F. was responsible for writing the manuscript text, programming the method, preparing the dataset, and implementing the idea. S.L., C.B. and Y.J. handled the preparation of the hardware. S.K. contributed by providing the database and operational scenario, conducting an in-depth discussion of the related

literature, and verifying the accuracy experiments exclusive to this paper. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work was supported by the 2023 Yeungnam University research grants. This research project was carried out with the support of the 2023 Daegu Mobile Robot Regulatory Free Zone innovation project “Demonstration of mobile cooperative robot for application of manufacturing process”, sponsored by the Ministry of SMEs and Startups (MSS) of the Korean government (grant no. P0023713).

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Restrictions apply to the availability of these data. Data was obtained from Pyeong Hwa Automotive Company and are available from the authors with the permission of Pyeong Hwa Automotive Company.

**Acknowledgments:** This work received support from Pyeong Hwa Automotive, with guidance provided by company employees Changjin Oh, Youngkyun Yoon, and Seungjae Oh for data preparation.

**Conflicts of Interest:** The authors declare no conflict of interest.

## Abbreviations

The following abbreviations are used in this manuscript:

|          |                                               |
|----------|-----------------------------------------------|
| ELAN     | extended efficient layer aggregation networks |
| YOLO     | you only look once                            |
| EGCY-Net | An ELAN and GhostConv-based YOLO Network      |
| CGStack  | Conv-GhostConv stack                          |
| YOLOR    | you only learn one representation             |
| mAP      | mean average precision                        |
| ONNX     | Open Neural Network Exchange                  |
| t-SNE    | t-distributed stochastic neighbor embedding   |

## References

1. Dorr, L.; Brandt, F.; Pouls, M.; Naumann, A. An Image Processing Pipeline for Automated Packaging Structure Recognition. In *Forum Bildverarbeitung*; KIT Scientific Publishing: Karlsruhe, Germany, 2020; p. 239.
2. Li, X.; Rao, W.; Lu, D.; Guo, J.; Guo, T.; Andriukaitis, D.; Li, Z. Obstacle Avoidance for Automated Guided Vehicles in Real-World Workshops Using the Grid Method and Deep Learning. *Electronics* **2023**, *12*, 4296. [\[CrossRef\]](#)
3. Mok, C.; Baek, I.; Cho, Y.S.; Kim, Y.; Kim, S.B. Pallet recognition with multi-task learning for automated guided vehicles. *Appl. Sci.* **2021**, *11*, 11808. [\[CrossRef\]](#)
4. Dörr, L.; Brandt, F.; Pouls, M.; Naumann, A. Fully-automated packaging structure recognition in logistics environments. In Proceedings of the 2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Vienna, Austria, 8–11 September 2020; Volume 1, pp. 526–533.
5. Liu, F.; Lu, Z.; Lin, X. Vision-based environmental perception for autonomous driving. *Proc. Inst. Mech. Eng. Part J. Automob. Eng.* **2022**. [\[CrossRef\]](#)
6. Liu, G.; Zhang, R.; Wang, Y.; Man, R. Road scene recognition of forklift agv equipment based on deep learning. *Processes* **2021**, *9*, 1955. [\[CrossRef\]](#)
7. Yan, N.; Chen, H.; Lin, K.; Li, Z.; Liu, Y. Fast and Effective Tag Searching for Multi-Group RFID Systems. *Appl. Sci.* **2023**, *13*, 3540. [\[CrossRef\]](#)
8. AI-Based Goods Recognition, Counting and Measuring. Available online: <https://www.logivations.com/en/> (accessed on 22 December 2023).
9. Vasileva, E.; Ivanovski, Z. A Hybrid CNN-DSP Algorithm for Package Detection in Distance Maps. *IEEE Access* **2023**, *11*, 113199–113216. [\[CrossRef\]](#)
10. Li, G.; Li, L.; Li, L.; Wang, Y.; Feng, B. Detection of containerized containers based on computer vision. In Proceedings of the 2021 IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), Chongqing, China, 12–14 March 2021; pp. 642–648.
11. Naumann, A.; Hertlein, F.; Dörr, L.; Furmans, K. Parcel3D: Shape Reconstruction from Single RGB Images for Applications in Transportation Logistics. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Vancouver, BC, Canada, 18–22 June 2023; pp. 4402–4412.



12. Zhou, Z.; Wang, M.; Chen, X.; Liang, W.; Zhang, J. Box Detection and Positioning based on Mask R-CNN [1] for Container Unloading. In Proceedings of the 2019 IEEE 4th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), Chengdu, China, 20–22 December 2019; pp. 171–174.
13. Dörr, L.; Brandt, F.; Naumann, A.; Pouls, M. TetraPackNet: Four-Corner-Based Object Detection in Logistics Use-Cases. In *DAGM German Conference on Pattern Recognition*; Springer International Publishing: Cham, Switzerland, 2021; pp. 545–558.
14. Wang, C.Y.; Bochkovskiy, A.; Liao, H.Y.M. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Vancouver, BC, Canada, 18–22 June 2023; pp. 7464–7475.
15. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv* **2017**, arXiv:1704.04861.
16. Skalski, P. ALPHA MAKE SENSE. Available online: <https://www.makesense.ai/> (accessed on 5 May 2023).
17. Han, K.; Wang, Y.; Xu, C.; Guo, J.; Xu, C.; Wu, E.; Tian, Q. GhostNets on heterogeneous devices via cheap operations. *Int. J. Comput. Vis.* **2022**, *130*, 1050–1069. [[CrossRef](#)]
18. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
19. Dollár, P.; Singh, M.; Girshick, R. Fast and accurate model scaling. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Virtual, 19–25 June 2021; pp. 924–932.
20. Tan, M.; Le, Q. Efficientnet: Rethinking model scaling for convolutional neural networks. In Proceedings of the International Conference on Machine Learning, Long Beach, CA, USA, 9–15 June 2019; pp. 6105–6114.
21. Radosavovic, I.; Kosaraju, R.P.; Girshick, R.; He, K.; Dollár, P. Designing network design spaces. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 14–19 June 2020; pp. 10428–10436.
22. Wang, C.Y.; Liao, H.Y.M.; Yeh, I.H. Designing network design strategies through gradient path analysis. *arXiv* **2022**, arXiv:2211.04800.
23. Redmon, J.; Farhadi, A. Yolo3: An incremental improvement. *arXiv* **2018**, arXiv:1804.02767.
24. Jocher, G.; Stoken, A.; Borovec, J.; Changyu, L.; Hogan, A.; Diaconu, L.; Poznanski, J.; Yu, L.; Rai, P.; Ferriday, R.; et al. ultralytics/yolov5:v3.0.Zenodo. 2020. Available online: <https://github.com/ultralytics/yolov5> (accessed on 10 January 2024)
25. Wang, C.Y.; Yeh, I.H.; Liao, H.Y.M. You only learn one representation: Unified network for multiple tasks. *arXiv* **2021**, arXiv:2105.04206.
26. Van der Maaten, L.; Hinton, G. Visualizing data using t-SNE. *J. Mach. Learn. Res.* **2008**, *9*, 2579–2605.

**Disclaimer/Publisher’s Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.