remote sensing

*Article*

# Domain Adaptation using Pre-trained Object Detection Models for Traffic Control Devices in Other Geographical Regions

**Dahyun Oh [1], Kyubyung Kang [2], Jinwu Xiao [2], Sungchul Seo[1], Kibum Kim[2], Hyungkeun Park[1,*], Junghoon Won[1]**

[1] Chungbuk National University, Department of Civil engineering; ekgus5484@gmail.com (D.O.); wow920308@gmail.com (S.S.); parkhk@chcungbuk.ac.kr (H.P.); jhwon@chungbuk.ac.kr (J.W.)
[2] Purdue University, School of Construction Management Technology; kyukang@purdue.edu (K.K.1); xiao270@purdue.edu (X.J.),
[3] Purdue University, Division of Construction Engineering and Management kim3854@purdue.edu (K.K.2)
* Correspondence: parkhk@chungbuk.ac.kr; Tel.:+82-043-249-1614

**Abstract:** Automated inspection systems for object detection using computer vision techniques can help manage traffic control devices(TCD) effectively. However, the main challenges of TCD detection models are the shortage of datasets for training and the difficulties in creating datasets due to the lack of research in different domains. To address this issue, this study provides a benchmark for using data from other relevant domains. Three model cases were developed in this study with different training methods: (1) training using COCO-based pre-training weights, (2) training using pre-trained weights from the source domain, and (3) training using a synthetic dataset of data from the source and target domains. The results of comparing the model cases show that source domain data cannot be directly applied to the target domain and that the pre-trained weights of the source domain help to build the target model efficiently. These results contribute a baseline for practitioners when building TCD models with a minimum of their own data.

**Keywords:** Domain Adaptation; Object Detection; Traffic Control Devices (TCD); Training Dataset Benchmark, YOLOv5;

**Citation:** To be added by editorial staff during production.

Academic Editor: Firstname Lastname

Received: date
Accepted: date
Published: date

## 1. Introduction

Computer vision technology has been employed in managing the built environment, such as inspecting and assessing conditions of civil infrastructure. In particular, previous research cases using deep learning-based computer vision technology to detect objects have proven to be highly accurate in the safety management of workers [1,2], construction equipment [3], and construction schedule management[4], so the frequency of use is expected to increase in the future [5]. Although computer vision studies have made tremendous progress in construction management, many researchers still need help constructing an ideal model for more accurate detection. The difficulty in building an ideal model is due to the problem of obtaining a highly qualified annotated dataset. The ideal model in this field is accurately localizing and recognizing objects in images or video frames [6]. That means that the model should be able to accurately distinguish target objects among a large variety of object categories and identify object instances from the same category, subject to intra-class variations [7]. To successfully build this ideal model, the model should learn the vast range of intra-class variations and a large number of object categories from the training process [7,8]. However, as described above, constructing this enormous dataset for training requires a higher labor-intensive and time-consuming, especially in the road or construction management areas with comprehensive coverage and high temporal variability.

There are two main representative approaches to solving the problem of building datasets from scratch and data shortage. The first approach uses similar object models or datasets already constructed in other regions or industries. For example, the Laboratory for Intelligent and Safe Automobiles (LISA) traffic signs dataset, a set of videos and annotated frames containing 47 types of U.S. traffic signs, is widely used in different research fields such as driver assistant and autonomous vehicles [9], traffic flow management, and road infrastructure [10]. In addition to this, there are also publicly available datasets, such as Microsoft Common Objects in Context (COCO) datasets. However, such a dataset has limitations in covering all road or construction sites with their unique characteristics.

In some cases, researchers create small-scale, individual datasets to achieve specific purposes and use them for research. Still, it is not easy for the datasets used in research to be shared publicly. The second approach employs a data augmentation technique. Data augmentation is a strategy to increase the training dataset by giving variations in the size and shape of the images while maintaining labels[11]. In construction management, this method is often applied to increase the variety of datasets [12-14] and balance the number of samples in different classes [15,16]. However, there is no reference to benchmark on how much data should be increased, and increasing the data does not necessarily lead to improved model performance [17,18].

Above all, it is worth noting that such approaches are not practical when applied to different domains. The performance of the object detection model is often affected by discrepancies in domain distribution. This tendency occurs when the model trained on a labeled dataset is applied to a dataset not seen in other models [19]. In particular, detecting traffic control devices is a field of research where domain shift by regions is likely to occur in the same category. Many traffic control device (TCD) systems have been tailored to specific traffic regulations in different regions. These two approaches do not provide a benchmark that can be used to answer the questions: (1) whether TCD practitioners can use pre-trained models, (2) whether they should develop their own training datasets and models, or (3) how many datasets need to be acquired and developed from their own domain. Hence, the objective of this study is to provide a benchmark that can be a foundation for automated maintenance and condition analysis for traffic control devices. In order to achieve the objective, this study conducted four steps which are (1) acquiring four datasets of different domains, (2) developing TCD detection models in the target domains, (3) evaluating the models for the use of pre-trained models and data from different geographical regions, and (4) reporting the benchmark of overall model performances. The provided benchmark shows that the mAP performance of the source model is degraded by 8% to 18% when it is applied directly to the new target domain. There is uncertainty about adopting a pre-trained model in other domains without enhancing data developed from the target domain. Therefore, stakeholders are encouraged to develop their own target domain data in conjunction with domain adaptation in other regions, even if it is a small target domain dataset. The benchmark in this study contains useful information for effectively building detection models in new domains, which helps practitioners plan for the resources needed to build detection models.

## 2. Related Studies

Traffic Control Devices (TCDs) are defined as all signs, signals, markings, and other devices placed along roads to promote road safety and efficiency by providing for the orderly movement of all road users. Especially in road construction zones, they need to be managed in a timely manner, as they warn drivers of dangerous situations in advance, thus preventing potential accidents around road construction. In the field of TCD object detection, most research has been conducted on studies related to autonomous driving and driver assistance systems, and they have focused on permanent devices such as traffic lights and traffic signs in TCDs. On the other hand, at road construction sites, TCDs are mainly used as temporary devices(TTCD) due to the characteristics of the construction site. Although maintenance is required for safety and traffic control efficiency, less research has focused on temporary devices used in road construction compared to permanent devices. As a result, few benchmark datasets for TTCD detection exist.

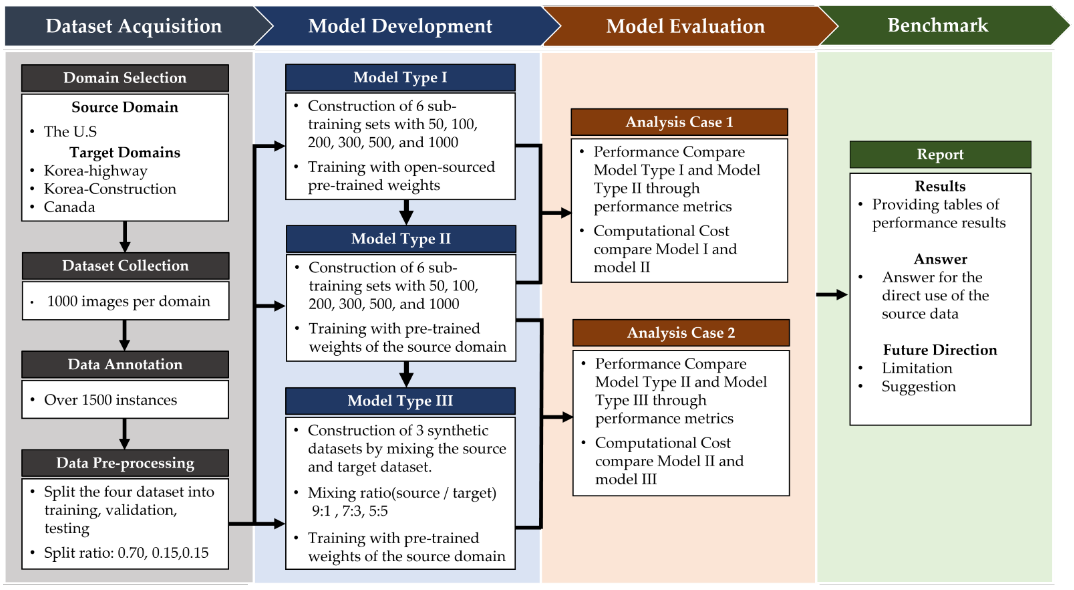
The research on computer vision for TCD has been studied for a decade. Stalkampet et al.[20] organized the benchmark competition for traffic sign detection and provided the results as the benchmark. The purpose of this competition was to provide the widespread benchmark called "The German Traffic Sign Recognition Benchmark," known as GTSRB. Timofte et al.[21] provided a multi-view scheme of a traffic sign dataset called a KUL Belgium Traffic Sign (KUL) benchmark to improve the efficiency of traffic sign detection and recognition. Larsson et al. [22] proposed to use locally segmented contours combined with an implicit star-shaped object model as a prototype for different sign classes to construct robust object models. They also provided publicly available benchmark datasets, Swedish Traffic Sign (STS) datasets, to demonstrate their proposal. Likewise, Mogelmose et al. [23] also introduced the effectively available traffic sign dataset in the U.S., the LISA dataset, through a comprehensive review of other traffic sign benchmarks. In particular, they pointed out the problem of developing a traffic sign detection model using the benchmarks constructed in other countries. According to their review, the designs of TCD systems are standardized through laws. Still, they differ across the World, so they addressed that using datasets from other countries is inefficient for developing a robust model. Zhu et al. [24] also introduced a Chinese traffic sign dataset called Tsinghua-Tencent 100K, which is more tailored to detect Chinese traffic signs according to the abovementioned problems.

In addition to these publicly available datasets, researchers often have developed and used a small number of them depending on their research purpose. For example, autonomous driving researchers [9,25,26] developed an object detection model using YOLO algorithms with their own datasets to detect different traffic cones. Dhall et al. [25] presented a method to detect traffic cones and estimate their real-time position in the 3D World using YOLOv2 algorithms. Albaráñez Martínez et al. [9] proposed a lightweight neural network to detect traffic cones on a racing car. In addition, Katsamenis et al. [26] introduced a new framework for cone detection by using the yolov5 algorithm and a manually created traffic cone image dataset from multiple sources. In addition to TCD detection research related to autonomous driving, research has also been conducted in maintenance, but it is scarce. Kang et al. [27] and Kim et al. [28] proposed an automated state analysis framework for pavement marking using the YOLOv3 algorithm. Seo et al. [29] and Song et al. [30] developed an object detection model for temporary traffic control devices using a YOLOv3 algorithm to manage the TTC devices in real-time.

Even though some studies have demonstrated that object detection techniques can help manage TCD effectively, TCD research remains under-explored regarding temporary devices. In order to fill this research gap, this study focuses on developing models in new target domains with as little data as possible. As other researchers have pointed out, using custom datasets is the best way to build robust models [11,21,24], but creating datasets from scratch can be challenging. In this respect, domain adaptation can be a solution. Domain adaptation techniques (DA) are a specific example of transfer learning (TL), which uses labeled data from one or more relevant other domains to perform similar tasks in the target domain. This technique enables the model to leverage the knowledge learned from prior tasks to increase prediction about a new task [31-34]. One of the main benefits of domain adaptation (or transfer learning) is the ability to train new models for a target task with much less labeled data [12,31,35-37]. Therefore, this study develops the model on a target domain with minimal data through domain adaptation techniques rather than building from scratch.

## 3. Methodology

This section explains how the study was conducted to achieve the objectives of the study. This research consists of four processes: dataset acquisition, model development, model evaluation, and benchmark. The dataset acquisition process includes selecting, collecting, annotating, and preprocessing data from different domains to create a benchmark dataset for each domain. The model development process involves developing three model types under different training methods. The model evaluation process is for comparing and analyzing the developed model cases. The last process is to report the key findings of the analysis results as a benchmark. The overall workflow of the study is shown in Figure 1.



Dataset Acquisition
Model Development
Model Evaluation
Benchmark
Domain Selection
Source Domain
• The U.S
Target Domains
• Korea-highway
• Korea-Construction
• Canada
Dataset Collection
• 1000 images per domain
Data Annotation
• Over 1500 instances
Data Pre-processing
• Split the four dataset into training, validation, testing
• Split ratio: 0.70, 0.15,0.15
Model Type I
• Construction of 6 sub-training sets with 50, 100, 200, 300, 500, and 1000
• Training with open-sourced pre-trained weights
Model Type II
• Construction of 6 sub-training sets with 50, 100, 200, 300, 500, and 1000
• Training with pre-trained weights of the source domain
Model Type III
• Construction of 3 synthetic datasets by mixing the source and target dataset.
• Mixing ratio(source / target) 9:1 , 7:3, 5:5
• Training with pre-trained weights of the source domain
Analysis Case 1
• Performance Compare Model Type I and Model Type II through performance metrics
• Computational Cost compare Model I and model II
Analysis Case 2
• Performance Compare Model Type II and Model Type III through performance metrics
• Computational Cost compare Model II and model III
Report
Results
• Providing tables of performance results
Answer
• Answer for the direct use of the source data
Future Direction
• Limitation
• Suggestion

*Figure 1 The Flowchart of the Study*

During the dataset acquisition, four targets of barrel datasets from the United States, Korea, and Canada were constructed. In this process, each dataset underwent collection, data annotation, and data preprocessing, respectively. In order to control the data systematically, each dataset is collected in a similar way and annotated with the same number of instances from the collected images. The constructed datasets were then split into training, validation, and testing for developing and evaluating the model. These datasets were denoted as Target S, Target A, Target B, and Target C for this study.

Target S dataset contains the U.S. barrel type that complies with the Manual on Uniform Traffic control devices (MUTCD) by FHWA [38]. According to MUTCD, the standard barrel size shall be at least 36 inches in height and 18 inches in width. Each barrel shall have a minimum of two oranges and two white retroreflective stripes, with the top stripe being oranges. The MUTCD does not specify the means of ballasting for barrels, but most of the barrels in this dataset were ballasted with rubber tire rings. Target A barrel complies with the Ministry of Land, Infrastructure, and Transport (MOLIT) Guidelines for Traffic Management at Road Construction Sites in Korea. In particular, this type is used in highway or freeway construction zones [39]. This type has a high degree of visual similarity to Target S in terms of size and color. The standard size of this type shall be 800 mm in height and 450 mm in width at least. The color of the barrel shall be orange with two white retrorelectorized stripes, as in the case of Target S. Target B barrel is another type used in Korea while complying with the MOLIT Guidelines. This type is typically used on routes with traffic lights or any short-term construction sites [39]. The general standards are the same as target A, but the noticeable difference is the shape. Unlike target A, target B is a separable assembly type with one wider white retrorelectorized stripe.

Target C barrel complies with the MUTCD for Canada published by The Transportation Association of Canada (TAC) [40]. Although most of the Canadian jurisdictions comply with the MUTCD, there are considerable differences in the design and regulations of TCDs between the provinces of Canada. For example, the barrel used in British Columbia, Canada's westernmost province, is almost identical to the U.S. However, according to Ontario Traffic Manual (OTM), the barrels should be used on freeways, other high speeds, and high volumes of roads [40]. In addition, the Ontario barrel has a height of 1000 mm and a minimum diameter of 360 mm for the bottom reflective band, which is taller and narrower than the British Columbia barrel. The most notable difference is the retroreflective stripes. The Ontario barrel must be horizontal and circumferential with alternating black and four retroreflective orange stripes. Since the Ontario barrel has a remarkable difference in design from the barrel that complied with MUTCD, it has a low visual similarity to the target S.

During model development, three types of models were developed using the Yolov5 algorithm. [41] Many researchers in object detection favor Yolov5 because of its fast inference time and high accuracy. [26,34,45-46] The framework of Yolov5 consists of three networks: backbone, neck, and output (shown in Figure 2(a)). Backbone networks are used to extract features from the input image by aggregating and forming image features at different granularities. The neck network is a series of layers to mix and combine image features before passing them to the prediction stage. As the neck network, Yolov5 uses both the Feature Pyramid Network (FPN) and the Path Aggregation Network (PAnet). The pyramid structure of FPN conveys powerful semantic features from the top feature map to the bottom feature map. Meanwhile, the pyramid structure of PANet conveys powerful localization features from the bottom feature map to the top feature map. The output network consists of three detection layers with feature maps of different scales of 80×80, 40×40, and 20×20 for detecting objects in the input image.

Moreover, the sub-layers were combined in the layers of networks such as C3 and SPSS(Shown in Figure 2(b)-(d)). The C3 layer is a cross-stage partial (CSP) bottleneck with three CONV(CBL) layers. CONV(CBL) layer is a combination of a standard convolutional CONV, batch normalization (B.N.), and the swish activation function of the Sigmoid linear unit (SiLU). The C3 layer is designed to improve the speed and weight by simplifying the original CSP bottleneck while maintaining the performance. [41] The spatial pyramid polling fast (SPPF) is a faster version of SPP. The SPP layer improves the receptive field by returning feature maps of arbitrary size to a fixed-size feature vector [42]. YOLOv5 also comprises different scales of models, such as YOLOV5s, YOLOv5m, YOLOv5l, and YOLOv5x, with alterations in depth and width in each model. Among these models, Yolov5l(Large) was chosen for this study because it has a deeper and wider network that enhances performance more [43-44] while meeting the workstation specifications.

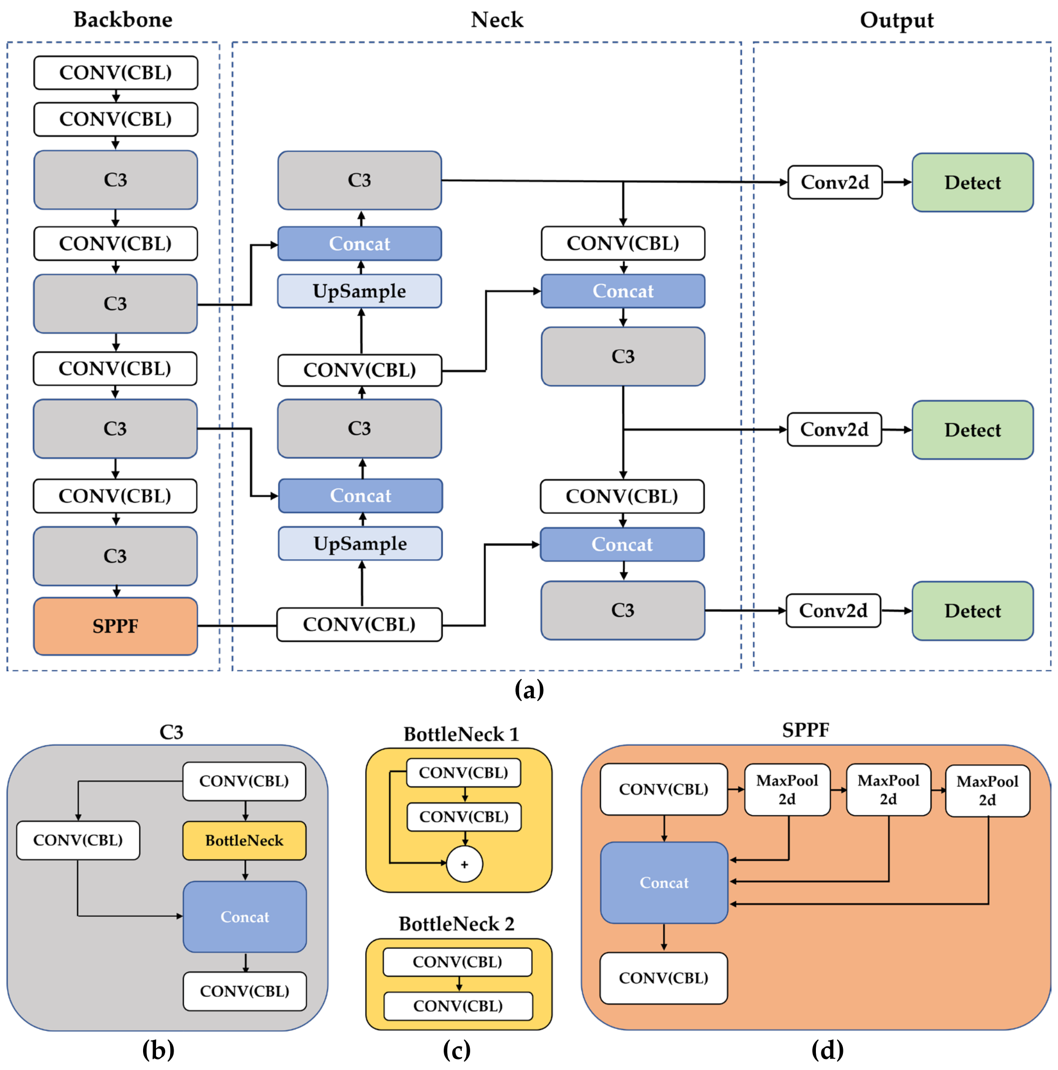


*Figure 2 The Framework of the YOLOv5 Structure (a) YOLOv5 networks: Backbone, Neck, Output. (b) C3 layer (c) BottleNecks in C3, 1 for Backbone and 2 for Neck. (d) SPPF layer*

Three model types are all developed based on the domain adaptation techniques but distinguished explicitly by their training methods. The first type is a general type that uses open-sourced pre-trained weights. The YOLOv5 models provide the pre-trained weights using the COCO dataset, which consists of 2,500,000 labeled instances of 91 common object classes.[47] Therefore, this study uses open-sourced pre-trained weights to build the first model type. In the process, this study also divided each training set into six sub-training sets with different amounts of instances to investigate the relationship between labeled data and model performance. The instances in each sub-training set were 50, 100, 200, 300, 500, and 1000, respectively. The second type of model was not much different from the first type. This type used the pre-trained weights of the source domain model instead of the pre-trained weights of YOLOv5. The last type of model also used the pre-trained weights of the source domain model when training. However, this type used a synthetic training set. A synthetic training set is a mixture of data from the source domain and data from the target domain. The four domains of datasets were randomly subsampled to create the synthetic training sets with different ratios. The three synthetic training sets contained 1000 instances each, and the mixing ratios of source and target data were 9:1, 7:3, and 5:5, respectively. In total, fifteen models were developed: six belonged to type I, six belonged to type II, and three belonged to type III.

Then, two analysis cases were conducted for model evaluation in the performance and cost evaluation process. The first analysis compared the performance of the model Type I and Type II. The second analysis case compared the performance of the model Type II and III. When comparing Type II and Type III, model type II was selected by matching the number of target instances with model type III. For example, when comparing a type III model trained with a synthetic dataset of 9 (900 instances) to 1 (100 instances) ratio, the selected model type II should be trained with 100 training instances. The overall model performance was evaluated using the four key metrics: precision, recall, F1 score, and mean average precision(mAP) [48-50]. These performance parameters are calculated based on the value of true positive (T.P.), false positive (F.P.), and false negative (F.N.). Each metric's mathematical expressions are displayed in equations (1) to (4). Precision represents the fraction of truly classified objects to the total number of objects classified by the model as the class. Recall indicates the ratio of the truly detected objects to the number of objects correctly predicted by the model.[48,49]

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive + False Positive}} \tag{1}$$

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive + False Negative}} \tag{2}$$

The high percentages of both precision and recall indicate that better model performance. This reason is that when the number of positive objects increases, the accuracy of correctly classifying each object decreases. Due to the importance of precision and recall, the two metrics should be balanced to optimize the model's performance. In order to measure the balance between precision and recall, the f1 score is used. The F1 score represents the harmonic mean of precision and recall[50]; thus, a higher F1- score indicates a better performance.

$$F_1\ \text{Score} = \frac{2\ \text{x Precision x Recall}}{\text{Precision + Recall}} \tag{3}$$

Then, all models are evaluated with mAP, calculated by averaging the average precision (A.P.) for all classes involved in the trained models. The higher mAP values reflect a well-performing model. [48-50]

$$\text{mAP} = \frac{1}{n}\sum_{k=1}^{k=n} AP_{\text{k}} \tag{4}$$

The computational cost of each model was also evaluated and compared by using training time, which is intended to provide a more efficient method when benchmarking model development. In particular, the overall performance difference (5) between the models is evaluated by percentage variance[51], which can describe the degree of improvement and degradation in the new model's performance compared to the previous model.

$$\text{Percentage variance} = \frac{\text{New model type – Old model type}}{\text{Old model type}}\ \text{x}\ 100 \tag{5}$$

After analyzing these models, the key findings of this study were extracted and summarized. The summarized vital findings would be a benchmark for practitioners who would like to utilize the source data when developing an object detection model in a new domain. Overall, all model types fit well in the target domain, despite the small amount of training data. In terms of computational efficiency, model type II outperformed the other types of the model, saving more than twice the training time than the other types. In the case of model III, it worked well in both the target and source domains, although it consumed a little more than model type II.

## 4. Benchmark

### *4.1 Dataset Acquisition*

Four construction barrel targets from the United States, Korea, and Canada were selected for this study. Selected barrels are identical in terms of the purpose of use, as they are temporary control devices used in work roads. However, they comply with different standards in shape, size, and layout under their traffic regulations. Such factors imply that domain shift occurs due to the country's different regulations, even if the TCD objects, mainly barrels in this study, are in the same category. Therefore, this study considers a barrel as a class and a domain as a region. In addition, other object detection researchers have already created the U.S. benchmark dataset [29]. Therefore, this study uses the existing U.S. benchmark dataset as the source domain and other datasets as target domains.

Each dataset was collected in a similar manner, although they were collected in different geographical locations. This study used a commercial camera as the data collection device. The cameras were mounted on the dashboard and recorded the road construction zone at 4k Ultra High Definition (UHD) 60 fps. Data were collected regardless of weather or time of day; therefore, the collected data varied in weather conditions, lighting intensity, and shadows.

The collected data was then annotated according to the labeling rules. For example, images per 15 consecutive frames were extracted from collected data to provide variation in object angles and lighting while reasonably trading off between numerous labeling data and labeling costs. Only barrels at a distance of 10-15 meters from the data collection vehicle were labeled to ensure clear visibility and readability.

The constructed datasets for each domain were: target S dataset consisted of 1000 images, target A dataset consisted of 657 images, target B consisted of 657 images, and target C consisted of 1,000 images. In addition, each image in these datasets was annotated with one or more instances. Four examples of a constructed dataset are displayed in Figure 3.

In the data preprocessing,1,500 instances were randomly extracted from each target benchmark dataset for analysis and comparison in the data preprocessing. The extracted instances were chosen as varied as possible to reduce the potential bias of data. After that, extracted instances were split into training, testing, and validation sets with a ratio of 0.70, 0.15, and 0.15, respectively.

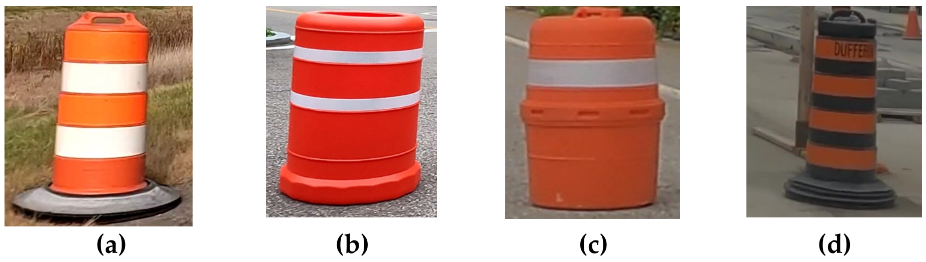


*Figure 3 Example of barrel dataset: (a) Target S (U.S barrel), (b) Target A (Korean barrel for highway), (c) Target B (Korean barrel for short-term construction), (d) Target C (Canada barrel)*

### *4.2 Model Development*

Three types of models were constructed to evaluate the use of the source data in the target domain. As described in Chapter 3, domain adaptation was applied to each model type to mitigate the generalization error due to insufficient data. Specifically, for type I, the models of six different training sets were trained with pre-training weights of the Yolov5-L model using the COCO dataset. The second model type applied the pre-trained weights of the source model trained with source data. The last model type was also trained with weights of the source model, but synthetic training sets of source and target data were used for the training dataset. In addition, for both types, I and II, the data in the training set were further divided into six training subsets with 50, 100, 200, 300, 500, and 1000 instances each. For type III, three sub-training sets were constructed by mixing the source and target datasets in different ratios to investigate the amount of labeled data required for the target domain when using the source data. Three mixing ratios of the source and target were at 9:1, 7:3, and 5:5, respectively, and the total number of instances was fixed at 1000 per each synthetic dataset. Table 1 summarizes the overall model types.

*Table 1 Summary of Model Types*

| **Types** | **Model I** | **Model II** | **Model III** |
|---|---|---|---|
| Training Dataset | Target | Target | Mixture (Source+Target) |
| # Training sets | 6 | 6 | 3 |
| Pre-trained Model | YOLOv5L using COCO | Source Domain | Source Domain |

Each model type was trained, validated, and tested in the 11.11 Pytorch framework, an open-source machine learning based on Python programming and the Torch library. The computational specifications of the workstation were Xeon W-2245 @3.9 GHz, Ubuntu 20.02, NVIDIA GeForce RTX 3090, and 128 GB of total memory. Each network of the models was trained with deep tuning, one of the TL approaches that allow modifying the weights of all layers of pre-trained networks when training the network for a new dataset [52]. Hyperparameters of the models were chosen to generally maintain their default settings and meet the running platform's hardware requirement. In detail, each network was trained using a stochastic gradient descent (SGD) optimizer with 0.0005 weight decay and 0.01 learning rate. All training was run with 300 epochs with a batch size of 32 at an image input size of 640 x 640.

### *4.3 Performance and Cost Evaluation*

Before developing three types of models, each target dataset was tested with an existing source model to investigate how well the source model worked in the domains of different countries. Table 2 shows the overall testing results of each target testing set without training. The existing source model performed well on the source dataset, showing high-performance metrics with a mAP of 0.985 and an F1-score of 0.958. However, when the source model was applied to a target dataset in other domains without training, the applied source model performed poorly, showing lower mAP values than the original. Significantly, the source model showed the worst mAP values when applied to target C, representing that the source domain's models are inferior when the visual similarity between the source domain and the target domain is relatively low.

*Table 2 Comparison of the Testing Results for Each Target Dataset without Training.*

| Testing set | Precision | Recall | F1-Score | mAP_0.5 [1] |
|---|---|---|---|---|
| Source S | 0.923 | 0.953 | 0.958 | 0.938 |
| Target A | 0.905 | 0.765 | 0.829 | 0.846 |
| Target B | 0.848 | 0.792 | 0.819 | 0.865 |
| Target C | 0.885 | 0.618 | 0.728 | 0.765 |

1 Mean Average Precision at 0.5 Intersection over Union (IoU) threshold.

### *4.3.1 Analysis Case 1 – Use of the pre-trained models.*

The performance of each target model was compared for two models, Type I and Type II. In both types, the mAP values of all target models improved with the number of labeled data in the training set. Among them, the target A dataset outperformed the other target datasets in both model types I and II. Figure 4 displays the relationship between the performance of the model and the amount of labeled data per target domain. Most of the model's performance improved significantly until the number of training instances reached 200. After 500 instances, the model's performance tended to converge to a certain level of accuracy.

As more data were added to the training set, the mAP values were less variable in Type II. In addition, the models achieved a certain level of accuracy even with small training data, with balanced recall and precision values. There were no systematic differences between the model types I and II until 500 data. However, after passing the number of 500 instances, all target models performed better when Type II was applied.

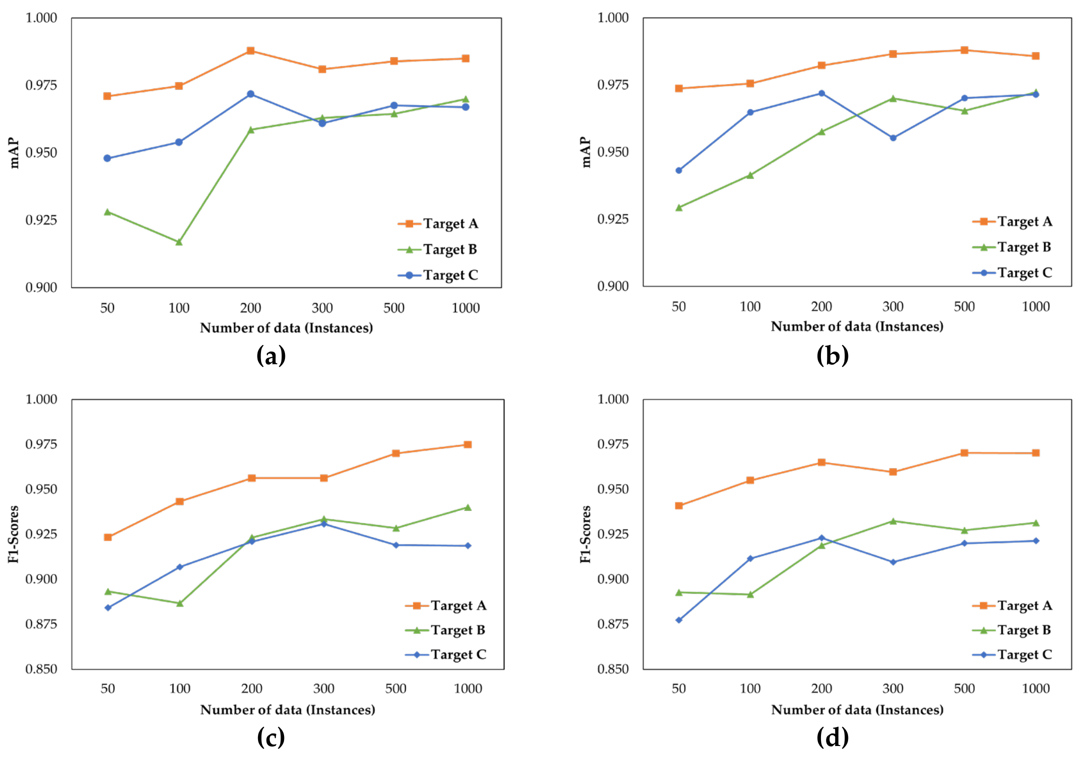


*Figure 4 Comparisons between model performance and the number of labeled instances per target domain. (a) mAP values of Model type I (b) mAP values under Model type II (c) F1-scores of Model type I (d) F-1 scores of Model type II.*

In terms of computational cost, there was a clear difference between the two types. In most cases, the computational costs were better in model type II (As shown in Table 3). Especially when the data were at 1000, this cost difference was considerable because the results showed that the computational cost was reduced by over 50% when applying model type II.

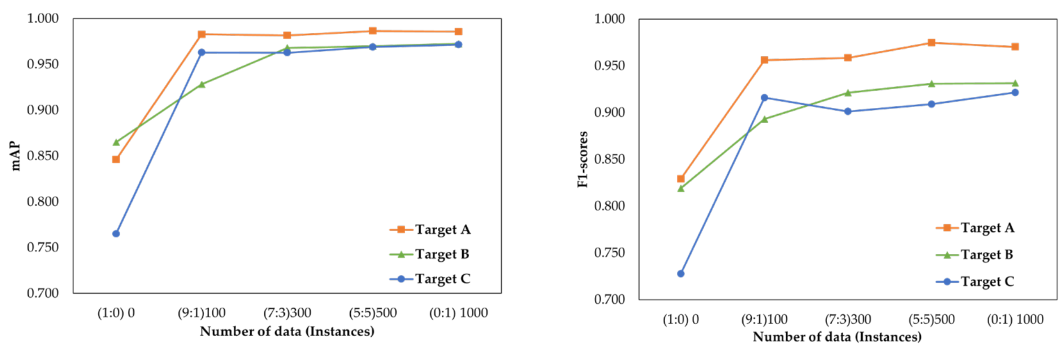
*Table 3 Percentage Variance of the Computational Cost of Model Types I and II*

| | **Percentage variance (%)**[2] | | |
|---|---|---|---|
| Labeled Instances | Target A | Target B | Target C |
| 50 | -8.7% | -17.1% | -5.6% |
| 100 | -27.7% | -97.5% | +0.1% |
| 200 | -2.7% | -0.8% | -42.1% |
| 300 | +1.4% | -0.2% | +1.7% |
| 500 | -0.2% | +0.6% | -1.7% |
| 1000 | **-57.9%** | **-156.6%** | **-98.7%** |

[2] Old model is Type I, and the new model is Type II

### 4.3.2 *Analysis Case 2 – Required target data*

The target data was mixed with the source data set to find the minimum amount of target data required. The model performance of the synthetic training dataset was then compared with the model trained with only the target dataset. Figure 5 shows the differences in model performance regarding different mixing ratios. Despite a small amount of target data mixed, the model's performance was significantly improved compared to the model trained with only source data. Although model performance did not vary significantly between the different mixing data sets per domain, mixing the data at a 1:1 ratio provided the best overall performance for the target model in both mAP and F1-Scores. Comparing the performance of the models by considering only the amount of used target data, there was no considerable difference between the two model types II and III.



**(a)** **(b)**

*Figure 5 Comparison between Model Performance and Mixed Target Data Ratios. (a) mAP Values of Datasets with Different Mixing Ratios (b) F1-scores of Datasets with Different Mixing Ratios*

As expected, model type II less took training time than type III. Even though both types of models used the same amount of target data, they have different amounts of the actual training data, as the number of training data for model type iii was fixed at 1000 instances. In Table 4, labeled instances represented the number of target data used in training.

*Table 4 Percentage Variance of the Computational Cost of Model Types II and III*

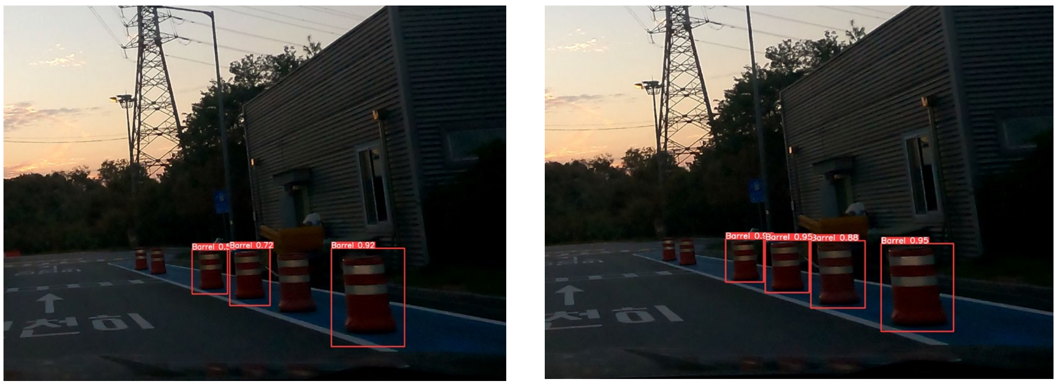
| | **Percentage Variance(%)**[3] | | |
|---|---|---|---|
| Labeled Instances | Target A | Target B | Target C |
| (9:1) 100 | 495% | 1560% | 849% |
| (7:3) 300 | 230% | 241% | 314% |
| (5:5) 500 | 17% | 138% | 253% |

[3] Old model is Type II, and the new model is Type III

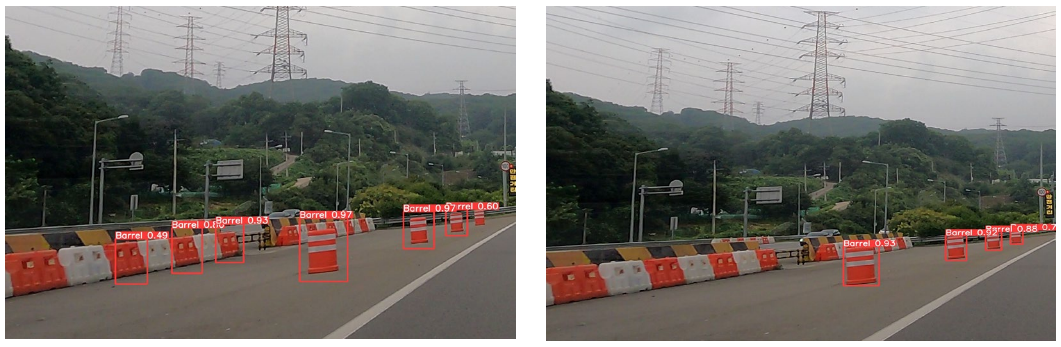
## 5. Discussion

Through experiments, this study determined how to effectively utilize the existing data from the source domain when developing an object detection model in a new target domain where no data is available. According to the untrained test results, model performance degraded when the model from the source domain was applied directly to the new target domain. This result is consistent with the phenomenon described in other domain adaptation studies [18] that occurs when the source and target domains are not the same. From these results, this study proved that even though the task of detecting TCD objects, specifically barrel objects in this study, is the same, the barrels have different domains based on each country. Moreover, our results showed that the performance of the source model decreased by about 18% in the target C domain, which has the lowest visual similarity to the source domain. That implies that the target C domain has a greater discrepancy than the source domain. Based on this result, it can be expected that when the visual similarity between domains is low, the discrepancy between them is greater.

Moreover, this study also visually observed how the source model degraded its performance by detecting testing sets. The source model detected fewer positive objects (T.P.), while the model trained with some target data detected more positive objects. Not only that, but the original source model also detected more negative objects (T.N.) as positive objects. Therefore, using the source model directly in the new target domain is risky because the original model does not work well in the new domain and performs poorly compared to working in the source domain. Examples of the detection results are displayed in Figure 6.



(a) (b)



**(c)** **(d)**

*Figure 6 Example of Detection Results. (a) Fewer T.P. Results of the Model Type I (b) More T.P. Results of the Model type II (c) More T.N. Results of the Model Type I (d) Less T.N. Results of the Model Type II*

This study also showed that the performance of all types of models improved significantly, even trained with a small amount of target data. Although good performance results were achieved for all types of models, Model II showed the best domain-adaptive performance at the best computational cost. This finding demonstrated that using training weights from similar target models from other domains helps to build a robust model and significantly reduces the training time. This result is consistent with previous studies that reusing knowledge of object features already learned from other models helps find the best weights quickly by sharing similar features. [53] In addition, model type II seems to be feasible only in the target domain because its performance degrades when applied to the source domain (as displayed in Table 5). Even though model type II was trained with the pre-trained weights of the source model, the mAP of the models of type II decreased significantly as the number of trained target data increased. In contrast, although model III requires a longer training time due to the amount of training data, it performed well on both the source and target domains. The mAP of the models only degrades by 2% maximum, which is negligible. Therefore, model type III seems to be interchangeable between the source and target domains.

*Table 5 Comparisons of Model Type II and III using the Source Domain's Test set*

| Model Type | Number of Trained data | mAP_0.5 | | | % mAP variance | | |
|---|---|---|---|---|---|---|---|
| | | Target A | Target B | Target C | Target A | Target B | Target C |
| Type II | 100 | 0.930 | 0.950 | 0.911 | -3% | -1% | -5% |
| | 300 | 0.841 | 0.900 | 0.872 | -14% | -6% | -10% |
| | 500 | 0.723 | 0.861 | 0.861 | -33% | -11% | -11% |
| Type III | (9:1) 100 | 0.954 | 0.963 | 0.961 | 0% | 1% | 0% |
| | (7:3) 300 | 0.958 | 0.952 | 0.941 | 0% | -1% | -2% |
| | (5:5 )500 | 0.953 | 0.948 | 0.951 | -1% | -1% | -1% |

This study also identified the minimum target data required for each domain by varying the amount of training data for each model type. Overall, the performance of the models improved with increasing the number of labeled data in the training set. This trend has been found in other studies in this field. [38,54-55] However, most similar studies were conducted at the image level rather than at the instance level. In addition, most of them covered only one domain. In the case of small datasets, even if each model has the same number of training images, the difference in the number of instances can significantly affect the model's performance. Also, the required training data might differ by domain, although the target classes are in the same categories. Therefore, this study addressed this issue by training different amounts of target data for each domain.

According to the results of this study (see Figure 4), most models' performance sharply increased until it reached 200 instances. After that, between the range of 200 and 500 instances, the performance of most models showed the highest variability. Especially in this range, most of the models' performance dropped. After passing this range, the performance values slightly increased or decreased and then converged. By this trend, the model's performance is expected to have an inflection point, typically a drop, before converging to a certain point. In addition, the higher the similarity between the target and source domains, the smaller the variation between the models' performances. As described in Figure b, the target A domain, the domain with the highest similarity to the source domain, has the lowest variance in the performance between the models. On the other hand, the target domain C, the domain with the lowest similarity to the source domain, fluctuated the performance significantly between the models. Despite this aspect, the model performance in this study fluctuated the least when the training instance was above 500 in most target domains.

## 6. Conclusions

Object detection using computer vision technology has the potential to significantly reduce practitioner workload through timely and frequent inspections in the field of TCDs management. This study provided a benchmark for building a TCD detection model in a new domain with minimum data. Three model cases were developed during model development by applying domain adaptation techniques that use pre-trained models trained in other related domains. Through comparing and analyzing three model cases, this study found that the model from the source domain cannot directly apply to the new target domain. Therefore, data from their own domain, even if it is small, is needed to ensure the model's performance. In addition, this study showed that training the model based on pre-trained weights from relevant domains can improve the model's performance and greatly increase the training speed. In addition, models trained with a small amount of target data and pre-trained weights from the source domain performed best in the target domain, but models trained with a synthetic dataset from the source and target domains performed well in both domains.

Although this study may be a useful benchmark for practitioners and researchers in a related field, it still has several limitations. First, this benchmark does not consider the various categories of TCD objects. Second, since this benchmark is intended for practitioners who want to model with less data, it does not consider data sets of various sizes. Finally, information about the use of more than one source domain is not considered. Therefore, further research is needed to establish better benchmarks that contribute to a more reliable automated checking system for future TCD management.

**Author Contributions:** Conceptualization, Oh., Seo, and Kang.; methodology, Oh., and Kang.; software, Oh., and Xiao.; validation, Oh.; formal analysis, Oh., Xiao., and Kang.; investigation, Oh., and Seo; resources, Oh., Seo, and Kang.; data curation, Oh., Seo., and Xiao; writing—original draft preparation, Oh.; writing—review and editing, Kim. and Kang.; visualization, Oh.; supervision, Kang, and Park.; project administration, Park and Won.; funding acquisition, Park and Won. All authors have read and agreed to the published version of the manuscript.

**Funding:** "This research was funded by BK21 program of Chungbuk National University."

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

## References

1. Fang, W.; Ding, L.; Zhong, B.; Love, P.; Luo, H. Automated detection of workers and heavy equipment on construction sites: A convolutional neural network approach, *Adv. Eng. Inform.* **2018**, 37, 139-149. doi: 10.1016/j.aei.2018.05.003

2. Delhi, V.; Sankarlal, R.; Thomas, A. Detection of personal protective equipment (PPE) compliance on construction site using computer vision based deep learning techniques, *Front. Built. Environ* **2020**, 6, 136. doi: 10.3389/fbuil.2020.00136
3. Chian, E.; Fang, W.; Goh, Y.; Tian, J. Computer vision approaches for detecting missing barricades, *Autom. Constr* **2021**, 131, 103862. doi: 10.1016/j.autcon.2021.103862
4. Yan, X.; Zhang, H. Computer vision-based disruption management for prefabricated building construction schedule, *J. Comput. Civ. Eng* **2021**, 35(6), 04021027.
5. Paneru, S.; Jeelani, I. Computer vision applications in construction: Current state, opportunities & challenges, *Autom. Constr* **2021**, 132, 103940. doi: 10.1016/j.autcon.2021.103940
6. Zaidi, S.; Ansari, M.; Aslam, A.; Kanwal, N.; Asghar, M.; Lee, B. A survey of modern deep learning-based object detection models, *Digital Signal Processing* **2022**, 103514. doi: 10.1016/j.dsp.2022.103514
7. Li, K.; Wan, G.; Cheng, G.; Meng, L.; Han, J. Object detection in optical remote sensing images: A survey and a new benchmark, *ISPRS J. Photogramm. Remote Sens* **2020**, 159, 296-307. doi:
8. Zhu, X.; Vondrick, C.; Ramanan, D.; Fowlkes, C.C. Do We Need More Training Data or Better Models for Object Detection. In: BMVC. **2012**.
9. Albaranez-Martinez, J.; Llopis-lbor, L.; Hernandez-Garcia, S.; Pineda de Luelmo, S.; Hernandez-Ferrandiz, D. A case of study on traffic cone detection for autonomous racing on a jetson platform, *IbPRIA* **2022**, Springer, Cham, 629-641. doi: 10.1007/978-3-031-04881-4_50
10. Su, Q.; Wang, H.; Xie, M.; Song, Y.; Ma, S.; Li, B.; Yang, Y.; Wang, L. Real-time traffic cone detection for autonomous driving based on YOLOv4. *IET intell. Transp.Syst.* **2022**, 16(10), 1380-1390. doi: 10.1049/itr2.12212
11. Zhuang, F.; Qi, Z.; Duan, K.; Xi, D.; Zhu, Y.; Zhu, H.; Xiong, H.; He, Q. A comprehensive survey on transfer learning. *In Proceedings of the IEEE* **2020**, 109(1), 43-76. doi: 10.1109/JPROC.2020.3004555.
12. Wang, M.; Deng, W. Deep visual domain adaptation: A survey. *Neurocomputing* **2018**, 312, 135-153. doi: 10.1016/j.neucom.2018.05.083.
13. Hsu, H.; Yao, C.; Tsai, Y.; Hung, W.; Tseng, H.; Singh, M.; Yang, M. Progressive domain adaptation for object detection. *WACV* **2020**, 738-746. doi: 10.1109/WACV45572.2020.9093358
14. Zhao, S.; Yue, X.; Zhang, S.; Li, B.; Zhao, H.; Wu, B.; Krishna, R.; Gonzalez, Joseph E.; Sangiovanni-Vincentelli, Alberto L.; Seshia, Sanjit A.; Keutzer, K. A review of single-source deep unsupervised visual domain adaptation. *IEEE Trans. Neural Netw. Learn. Syst* **2020** , 33(2), 473-493. doi: 10.1109/TNNLS.2020.3028503
15. Tseng, H.; Lee, H.; Huang, J.; Yang, M. Cross-domain few-shot classification via learned feature-wise transformation. *arXiv* **2020**, 2001, 08735, doi: 10.48550/arXiv.2001.08735
16. Kim, Y.; Cho, D.; Han, K.; Panda, P.; Hong, S. Domain adaptation without source data. *IEEE TAI* **2021**, 2(6), 508-518. doi: 10.1109/TAI.2021.3110179.
17. Na, S.; Heo, S.; Han, S.; Shin, Y.; Lee, M. Development of an artificial intelligence model to recognize construction waste by applying image data augmentation and transfer learning, *Buildings* **2022**, 12(2), 175. doi: 10.3390/buildings12020175
18. Wang, Z.; Yang, J.; Jiang, H.; Fan, X. CNN training with twenty samples for crack detection via data augmentation, *Sensors* **2020**, 20(17), 4849. doi: 10.3390/s20174849
19. Chen, Y.; Liu, Q.; Wang, T.; Wang, B.; Meng, X. Rotation-Invariant and Relation-Aware Cross-Domain Adaptation Object Detection Network for Optical Remote Sensing Images. *Remote Sensing* 2021, 13(21), 4386. doi: 10.3390/rs13214386
20. Stallkamp, J.; Schlipsing, M.; Salmen, J.; Igel, C. The German Traffic Sign Recognition Benchmark: A multi-class classification competition, *The 2011 International Joint Conference on Neural Networks* **2011**, pp. 1453-1460, doi: 10.1109/IJCNN.2011.6033395
21. Timofte, R.; Zimmermann, K.; Van Gool, L. Multi-view traffic sign detection, recognition, and 3D localisation. *Machine vision and applications* **2014**, 25(3), 633-647. doi: 10.1007/s00138-011-0391-3
22. Larsson, F.; Felsberg, M. Using Fourier Descriptors and Spatial Models for Traffic Sign Recognition. In *Image Analysis SCIA* **2011**, 2nd ed.; Heyden, A., Kahl, F. Eds.; Springer, Berlin, Heidelberg, **2011**; Volume 6688, pp. 238-249. doi:10.1007/978-3-642-21227-7_23
23. Mogelmose, A.; Trivedi, M.; Moeslund, T. Vision-based traffic sign detection and analysis for intelligent driver assistance systems: Perspectives and survey. *IEEE T-ITS* **2012,** 13(4), 1484-1497. doi: 10.1109/TITS.2012.2209421
24. Zhu, Z.; Liang, D.; Zhang, S.; Huang, X.; Li, B.; Hu, S. Traffic-Sign Detection and Classification in the Wild, *IEEE CVPR*, **2016**, 2110-2118, doi: 10.1109/CVPR.2016.232.
25. Dhall, A.; Dai, D.; Van Gool, L. Real-time 3D Traffic Cone Detection for Autonomous Driving. *IEEE IV* **2019**, 494-501, doi: 10.1109/IVS.2019.8814089
26. Katsamenis, I.; Karolou, E.E.; Davradou, A.; Protopapadakis, E.; Doulamis, A.; Doulamis, N.; Kalogeras, D. TraCon: A Novel Dataset for Real-Time Traffic Cones Detection Using Deep Learning. *In NiDS* **2022**. 2nd ed.; Krouska, A., Troussas, C., Caro, J. Eds.; Springer, Cham, **2022**, Volume 556, pp. 382-391. doi: 10.1007/978-3-031-17601-2_37
27. Kang, K.; Chen, D.; Peng, C.; Koo, D.; Kang, T.; Kim, J. Development of an automated visibility analysis framework for pavement markings based on the deep learning approach. *Remote Sensing* **2020**., 12(22), 3837. doi: 10.3390/rs12223837
28. Kim, Y.; Song, K.; Kang, K. Framework for Machine Learning-Based Pavement Marking Inspection and Geohash-Based Monitoring. *ICTD* **2022,** 123-132, doi: 10.1061/9780784484k319.012
29. Seo, S.; Chen, D.; Kim, K.; Kang, K.; Koo, D.; Chae, M.; Park, H. Temporary traffic control device detection for road construction projects using deep learning application. *In Construction Research Congress* **2022**. pp. 392-401

30. Song, K.; Chen, D.; Seo, S.; Jeon, J.; Kang, K. Feasibility of Deep Learning in Segmentation of Road Construction Work Zone using Vehicle-Mounted Monocular Camera. *UKC* **2021**, 15-18.
31. Csurka, G. Domain adaptation for visual applications: A comprehensive survey. *arXiv preprint* **2017** doi: arXiv:1702.05374.
32. Yosinski, J.; Clune, J.; Bengio, Y.; Lipson, H. How transferable are features in deep neural networks?. *Adv Neural* **2014**, 27
33. Shahinfar, S., Meek, P., and Falzon, G. How many images do I need?" Understanding how sample size per class affects deep learning model performance metrics for balanced designs in autonomous wildlife monitoring. *Ecological Informatics* **2020**., 57, 101085.
34. Sharma, T.; Debaque, B.; Duclos, N.; Chehri, A.; Kinder, B.; Fortier, P. Deep Learning-Based Object Detection and Scene Perception under Bad Weather Conditions. *Electronics* **2022**., 11(4), 563. doi: 10.3390/electronics11040563
35. Guo, Y.; Shi, H.; Kumar, H.; Grauman, K.; Rosing, T.; Feris, R. SpotTune: Transfer Learning Through Adaptive Fine-Tuning. *IEEE CVPR* **2019**, pp. 4800-4809, doi: 10.1109/CVPR.2019.00494.
36. Misra, I.; Shrivastava, A.; Gupta, A.; Hebert, M. Cross-Stitch Networks for Multi-task Learning. *IEEE CVPR*, **2016**, pp. 3994-4003, doi: 10.1109/CVPR.2016.433.
37. Bengio, Y. Deep learning of representations for unsupervised and transfer learning. *In: Proceedings of ICML workshop on unsupervised and transfer learning*. JMLR Workshop and Conference Proceedings, **2012**. p. 17-36.
38. Federal Highway Administration. Manual on uniform traffic control devices (MUTCD). **2009**
39. Ministry of Land, Infrastructure and Transportation (MOLIT), Traffic management guidelines for road 25 construction sites, MOLIT, Sejong, Korea, **2018**.
40. Manual, Ontario Traffic (OTM). Book 7: Temporary Conditions. Ontario Ministry of Transportation, St. Catherines, Ontario, Canada **2014**.
41. Jocher, G. YOLOv5 by Ultralytics, **2020** [Computer software]. https://doi.org/10.5281/zenodo.3908559
42. He, K.; Zhang, X.; Ren, S.; Sun, J. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. *IEEE TPAMI* **2015**, 37(9), 1904-1916, doi: 10.1109/TPAMI.2015.2389824.
43. Golubeva, A., Neyshabur, B., and Gur-Ari, G. Are wider nets better given the same number of parameters? *arXiv preprint* **2020**. doi: arXiv:2010.14495.
44. Liu, Y.; He, G.; Wang, Z.; Li, W.; Huang, H. NRT-YOLO: Improved YOLOv5 Based on Nested Residual Transformer for Tiny Remote Sensing Object Detection, *Sensors* **2022**, *22*, 4953. https://doi.org/10.3390/s22134953
45. Wang, J.; Chen, Y.; Dong, Z.; Gao, M. Improved YOLOv5 network for real-time multi-scale traffic sign detection. *Neural Comput & Applic* **2022**., 1-13 doi: 10.1007/s00521-022-08077-5
46. Zhu, Y.; Yan, W.Q. Traffic sign recognition based on deep learning. *Multimed Tools Appl* **2022**,81, 17779–17791 doi: 10.1007/s11042-022-12163-0
47. Lin, TY. et al. (2014). Microsoft COCO: Common Objects in Context. *Computer Vision ECCV* **2014**. In Lecture Notes in Computer Science, 2nd ed.; Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. Eds.; Springer, Cham. 2014, volume 8693. doi: 10.1007/978-3-319-10602-1_48
48. Padilla, R.; Passos, W.L.; Dias, T.L.B.; Netto, S.L.; da Silva. A Survey on Performance Metrics for Object-Detection Algorithms. *IEEE IWSSIP* **2020**, pp. 237-242, doi: 10.1109/IWSSIP48289.2020.9145130.
49. Zhang, X.; Wang, W.; Zhao, Y.; Xie, H. An improved YOLOv3 model based on skipping connections and spatial pyramid pooling. *Syst. Sci. Control Eng* **2021**, 9, pp. 142-149, doi: 10.1080/21642583.2020.1824132.
50. Padilla, R.; Passos, W.L.; Dias, T.L.B.; Netto, S.L.; da Silva, E.A.B. A Comparative Analysis of Object Detection Metrics with a Companion Open-Source Toolkit. *Electronics* **2021**, 10, 279. doi: 10.3390/electronics10030279
51. Brechner, R.; Bergeman, G. Contemporary Mathematics for Business & Consumers. *Cengage Learning*, **2016**.
52. N. Tajbakhsh N.; Shin, J.; Gurudu, S.; Hurst, R.; Kendall, C.; Gotway, M.; J, L. Convolutional Neural Networks for Medical Image Analysis: Full Training or Fine Tuning? *IEEE Trans Med Imaging*, **2016,** 35(5), 1299-1312, doi: 10.1109/TMI.2016.2535302.
53. Gayakwad, E.; Prabhu, J.; Anand, R.V.; Kumar; M.S. Training Time Reduction in Transfer Learning for a Similar Dataset Using Deep Learning. In Intelligent Data Engineering and Analytics. 2nd ed.; Satapathy, S.; Zhang, YD.; Bhateja, V.; Majhi, R. Eds.; *Adv. Intell. Syst. Compu* **2021**. Springer: Singapore, Volume 1177. Pp 359-367 doi: 10.1007/978-981-15-5679-1_33.
54. Sun, C.; Shrivastava, A.; Singh, S.; Gupta, A. Revisiting Unreasonable Effectiveness of Data in Deep Learning Era. *IEEE ICCV* **2017**, 843-852. doi: 10.1109/ICCV.2017.97.
55. Tabak, M. A., Norouzzadeh, M. S., Wolfson, D. W., Sweeney, S. J., VerCauteren, K. C., Snow, N. P., Miller, R. S. Machine learning to classify animal species in camera trap images: Applications in ecology. *Methods in Ecology and Evolution* **2019**. 10(4), 585-590.