remote sensing

Article

Low-Cost Object Detection Models for Traffic Control Devices through Domain Adaption of Geographical Regions

Dahyun Oh [1], Kyubyung Kang [2], Sungchul Seo [1], Jinwu Xiao [2], Kyochul Jang [3], Kibum Kim [4], Hyungkeun Park [1,*] and Jeonghun Won [5]

1 Department of Civil Engineering, Chungbuk National University, Cheongju-si 28644, Republic of Korea
2 School of Construction Management Technology, Purdue University, West Lafayette, IN 47907, USA
3 Department of Computer Science, Purdue University, West Lafayette, IN 47907, USA; jang128@purdue.edu
4 Division of Construction Engineering and Management, Purdue University, West Lafayette, IN 47907, USA
5 Department of Safety Engineering, Chungbuk National University, Cheongju-si 28644, Republic of Korea
* Correspondence: parkhk@chungbuk.ac.kr; Tel.: +82-043-249-1614

Abstract: Automated inspection systems utilizing computer vision technology are effective in managing traffic control devices (TCDs); however, they face challenges due to the limited availability of training datasets and the difficulty in generating new datasets. To address this, our study establishes a benchmark for cost-effective model training methods that achieve the desired accuracy using data from related domains and YOLOv5, a one-stage object detector known for its high accuracy and speed. In this study, three model cases were developed using distinct training approaches: (1) training with COCO-based pre-trained weights, (2) training with pre-trained weights from the source domain, and (3) training with a synthesized dataset mixed with source and target domains. Upon comparing these model cases, this study found that directly applying source domain data to the target domain is unfeasible, and a small amount of target domain data is necessary for optimal performance. A model trained with fine-tuning-based domain adaptation using pre-trained weights from the source domain and minimal target data, proved to be the most resource-efficient approach. These results contribute valuable guidance for practitioners aiming to develop TCD models with limited data, enabling them to build optimal models while conserving resources.

Keywords: domain adaptation; low-cost object detection; traffic control devices (TCDs); training dataset benchmark; YOLOv5

check for updates

Citation: Oh, D.; Kang, K.; Seo, S.; Xiao, J.; Jang, K.; Kim, K.; Park, H.; Won, J. Low-Cost Object Detection Models for Traffic Control Devices through Domain Adaption of Geographical Regions. *Remote Sens.* **2023**, *15*, 2584. https://doi.org/10.3390/rs15102584

Academic Editors: Riccardo Roncella and Mattia Previtali

Received: 17 April 2023
Revised: 8 May 2023
Accepted: 12 May 2023
Published: 15 May 2023

1. Introduction

Computer vision technology has been employed in managing the built environment, such as inspecting and assessing the condition 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 in 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]. This 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 more labor-intensive and time-consuming process, 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 assistance and autonomous vehicles [9], traffic flow management, and road infrastructure [10]. In addition to this, there are also publicly available datasets, such as the 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. Nonetheless, 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 providing 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 how much data should be added, and increasing the data does not necessarily lead to improved model performance [17,18]. In general, the stability and performance of a model are improved with a reduction in variance in the data. According to mathematical statistics, the law of large numbers and the central limit theorem suggest that if the sample size is sufficiently large, the data will tend towards a standard normal distribution that reflects the population. Consequently, if the variance of the existing data is substantial and can be reduced through data augmentation, increasing the number of data samples will result in better model performance. However, if the variance of the data is increased by incorporating entirely different data, the performance of the model will deteriorate. In conclusion, it is not always the case that a larger data sample leads to improved model performance.

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]. This is particularly relevant in the context of traffic control device (TCD) detection, as TCD systems are often tailored to specific regional traffic regulations. Despite this, current TCD object detection datasets and models are primarily developed using data from advanced AI countries, such as the United States, China, and Europe, rather than being tailored to specific countries or regions. Countries in the early stages of AI development, such as South Korea, Canada, and India, are also striving to develop their own training datasets for the development of object detection models. However, due to a lack of sufficient resources and infrastructure, they face challenges in building datasets that are appropriate for local contexts. To address this issue, the objective of this research is to propose a benchmark for efficiently utilizing traffic control device (TCD) data from multiple countries, with the goal of reducing the resources required for data collection, annotation, and training in the early stages of TCD object detection model development. In order to achieve the research objective, this study was conducted in four steps, which were (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 performance. 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 the 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 number of target domain dataset. The key contributions of this study can be summarized as follows:

1. It provides guidance on the amount of data required for efficient model training to achieve a specific level of accuracy for researchers facing a similar situation to this study (i.e., building a new object detection model, but importing and utilizing data from a model built in another country). This guidance can help prevent unnecessary wastage of resources (time and money) by providing researchers with the appropriate amount of data needed for effective model training, promoting active research activities in the field.
2. This study derived multiple condition-specific reference values for the TTCD domain, taking into account different design specifications of the barrel across different nations, use of different pre-trained weights, and different numbers of training datasets. These values can guide researchers in obtaining the appropriate amount of data required for effective model training, further enhancing the accuracy and effectiveness of object detection models in the TTCD domain.

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, TCDs 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 TCDs such as traffic lights and traffic signs. On the other hand, at road construction sites, TCDs are mainly used as temporary devices (TTCDs) 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.

Research on computer vision for TCDs has been conducted for a decade. Stallkamp 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 the 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, the Swedish Traffic Sign (STS) datasets, to demonstrate their proposal. Likewise, Mogelmose et al. [23] also introduced an 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 benchmarks constructed in other countries. According to their review, the designs of TCD systems are standardized through laws. However, they differ across the world, so the authors addressed these differences by using datasets from other countries, but they were proven to be inefficient in developing a robust model. Zhu et al. [24] also introduced a Chinese traffic sign dataset called Tsinghua-Tencent 100 K, which is more tailored to the detection of 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 their own 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 positions 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.

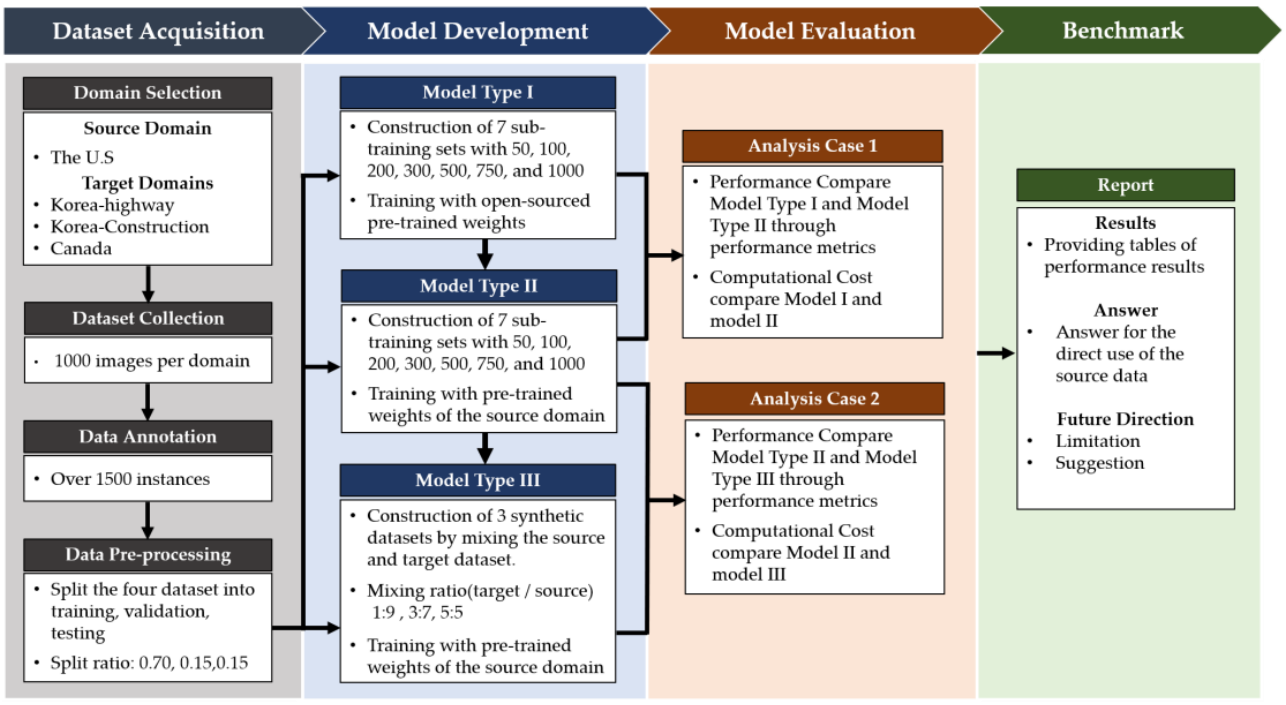
Although some studies have demonstrated that object detection techniques can help to manage TCDs effectively, TCD research remains under-explored regarding temporary devices. In order to fill this research gap, this study focuses on developing models in new target domains with as little data as possible. 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 type of transfer learning that uses labeled data from other relevant domains to perform similar tasks in the target domain. By leveraging prior knowledge [31–34], domain adaptation enables models to be trained with much less labeled data [12,31,35–37]. In general, this approach assumes that the source and target domains share the same task, i.e., Ts = Tt, and any differences between the datasets are caused only by domain divergence, i.e., Ds $\neq$ Dt, in the data distribution or feature space. [12]

Domain adaptation has various approaches depending on the specific settings and problems involved. In the early years of domain adaptation research, the focus was on shallow domain adaptation techniques, which include instance-based [38,39] and feature-based methods [40,41]. These techniques were developed to reduce the domain discrepancy between the source and target datasets by either reweighting source samples or learning a shared space that matches the distributions of the two datasets. Deep domain adaptation leverages deep neural networks to learn complex and abstract features via back-propagation, enabling more effective knowledge transfer between source and target domains. Common deep domain adaptation techniques include fine-tuning deep neural networks, GAN-based adversarial approaches [42], and data-reconstruction [43], which uses deep encoder–decoder architectures to ensure learned features between domains are invariant or shared. The principle of minimizing domain discrepancy is shared among these techniques and aligns with the aim of domain adaptation.

Domain adaptation techniques are widely employed to address data scarcity issues across various fields. Rostami et al. [44] proposed a semi-supervised domain adaptation framework for classifying Synthetic Aperture Radar (SAR) images with limited labeled data by transferring knowledge from the related Electro-Optical (EO) domain through a shared invariant cross-domain embedding space learned by coupled deep neural networks. Wang et al. [45] developed a Joint Coral-based Graph Neural Network (JCGNN) for unsupervised domain adaptation in multitemporal, hyperspectral remote sensing images, effectively using spectral relational information and domain-level and class-level distribution adaptation with GNN and class-wise CORAL methods. Lasloum et al. [46] presented a multi-source semi-supervised domain adaptation method (SSDAN) for remote sensing scene classification using EfficientNet-B3 CNN and a novel minimax entropy loss function, achieving remarkable performance with few labeled target samples. Zheng et al. [47] introduced a novel framework called Domain Adaptation via Task-Specific Classifier (DATSNET) to tackle the data shortage problem in high spatial resolution (HSR) imagery scene classification by aligning feature distributions between source and target domains, ultimately enhancing cross-scene classification performance. Although the domain adaptation field actively addresses data scarcity issues, most research focuses on algorithm improvement, with little emphasis on determining the minimum amount of target domain data needed for optimal results when using data from other related domains. Consequently, this study aims to identify the minimum data requirements in the target domain and explore cost-effective strategies using deep domain adaptation-based fine-tuning of deep neural networks, leveraging data from different domains for optimal results.

3. Methodology

This section explains how this study was conducted to achieve the objectives of the research. This study consisted of four processes: dataset acquisition, model development, model evaluation, and benchmark. The dataset acquisition process included selecting, collecting, annotating, and preprocessing data from different domains to create a benchmark dataset for each domain. The model development process involved developing three model types under different training methods. The model evaluation process aimed at comparing and analyzing the developed model cases. The last process was to report the key findings of the analysis results as a benchmark. The overall workflow of this 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 7 sub-training sets with 50, 100, 200, 300, 500, 750, and 1000
• Training with open-sourced pre-trained weights
Model Type II
• Construction of 7 sub-training sets with 50, 100, 200, 300, 500, 750, 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(target / source) 1:9 , 3:7, 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 this 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. In order to control the data systematically, each dataset was 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 [48]. According to the MUTCD, the standard barrel size should be at least 36 inches in height and 18 inches in width. Each barrel should have a minimum of two orange and two white retroreflective stripes, with the top stripe being orange. 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 [49]. This type has a high degree of visual similarity to Target S in terms of size and color. The standard size of this type is at least 800 mm in height and 450 mm in width. The color of the barrel should be orange, with two white retroreflective stripes, as in the case of Target S. The 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 [49]. The general standards are the same as for Target A, but a noticeable difference is the shape. Unlike Target A, Target B is a separable assembly type with one wider white retroreflective stripe.

The Target C barrel complies with the MUTCD for Canada, published by the Transportation Association of Canada (TAC) [50]. 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 that used in the U.S. However, according to the Ontario Traffic Manual (OTM), the barrels should be used on freeways and other high-speed and high-volume roads [50]. In addition, the Ontario barrel has a height of 1000 mm and a minimum diameter of 360 mm for the bottom reflective band, making it 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 four alternating black and retroreflective orange stripes. Since the Ontario barrel has a remarkable difference in design from the barrel that complies with the MUTCD, it has low visual similarity to Target S.

During model development, three types of models were developed using the YOLOv5 algorithm [51]. Many researchers in object detection favor YOLOv5 because of its fast inference time and high accuracy [26,34,52,53]. The framework of YOLOv5 consists of three networks: backbone, neck, and output (shown in Figure 2a). 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 used 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 2b–d). The C3 layer is a cross-stage partial (CSP) bottleneck with three CONV(CBL) layers. The 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 [51]. 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 [54]. YOLOv5 also comprises different scales of models, such as YOLOV5n, 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 meets the workstation specifications while having a sufficiently deep and wide network to further improve the performance [55,56].

In addition to its network architecture, YOLOv5 employs three loss functions (1) during training to enhance performance. These loss functions include localization loss, objectiveness loss, and the classification loss, represented as:

$$\text{Loss} = L_{loc(CIoU)} + L_{obj} + L_{cls} \tag{1}$$

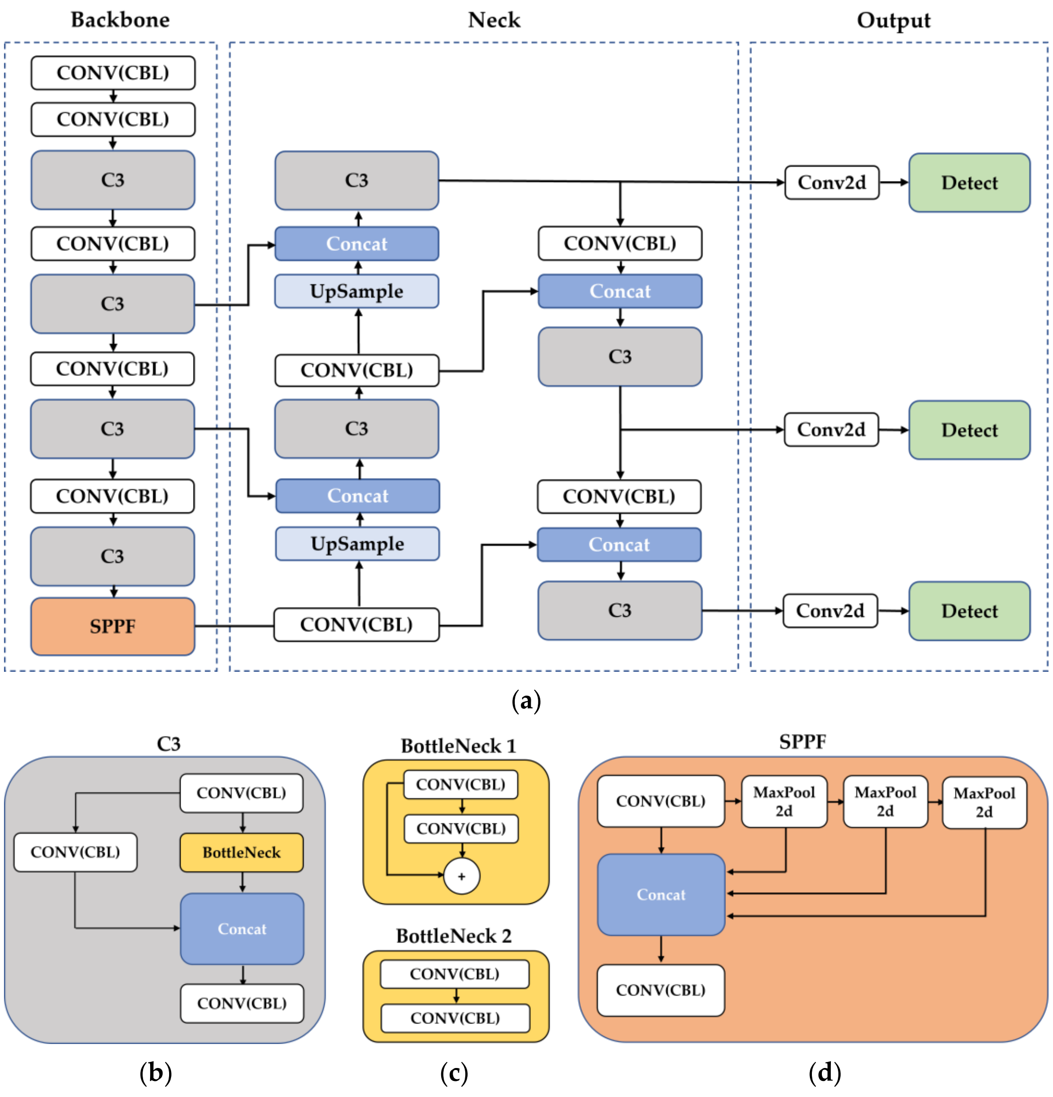


Figure 2. The framework of the YOLOv5 structure: (**a**) YOLOv5 networks: backbone, neck, output; (**b**) C3 layer; (**c**) bottlenecks in C3—1 for backbone and 2 for neck; (**d**) SPPF layer.

The localization loss aims to optimize the detection of the center point (x and y) as well as its width (w) and the height (h). Unlike YOLOv3 model, YOLOv5 improves localization loss by replacing mean squared error (MSE) with Complete Intersection over Union (CIoU) [57]. This approach effectively optimizes regression loss by simultaneously taking into account the overlapping area, the distance between center points, and the aspect ratio of the bounding box. The detailed localization loss function is as follows:

$$L_{loc(CIoU)} = 1 - IoU + \frac{\rho^2(b, b^{gt})}{c^2} + \alpha v \tag{2}$$

$$\alpha = \frac{v}{(1 - IoU) + v} \tag{3}$$

$$v = \frac{4}{\pi^2}\left(arctan\frac{w^{gt}}{h^{gt}} - arctan\frac{w}{h}\right)^2 \tag{4}$$

$$L_{obj} = -\sum_{i=0}^{S^2}\sum_{j=0}^{B} 1_{ij}^{obj}\left[\widehat{C}_l \log(C_i) + \left(1-\widehat{C}_l\right)\log(1-C_i)\right] \\ -\lambda_{noobj}\sum_{i=0}^{S^2}\sum_{j=0}^{B} 1_{ij}^{noobj}\left[\widehat{C}_l \log(C_i) + \left(1-\widehat{C}_l\right)\log(1-C_i)\right] \tag{5}$$

$$L_{cls} = -\sum_{i=0}^{S^2} 1_{ij}^{obj} \sum_{c \in classes} \left[\widehat{p}_l(c)\log(p_i(c)) + (1-\widehat{p}_l(c))\log(1-p_i(c))\right] \tag{6}$$

where ρ^2 represents the Euclidean distance between the center points of the predicted box b and the ground truth b^{gt}. The variable c is the diagonal distance of the smallest bounding box that completely encloses both predicted and ground truth boxes. α is a positive compromise parameter that gives priority to overlap over non-overlap, and υ measures the consistency of the aspect ratio between the two bounding boxes.

The objectiveness loss (5), also known as confidence loss, is used to determine the presence of a target object in the allocated grid cell. Classification loss, on the other hand, is used for accurate object class detection. Both objectiveness loss and the classification loss utilize binary cross entropy, in line with the YOLOv3. The formula for these two-loss function are as follows:

The S^2 grid cells represent S $\times$ S grids, with each cell generating B candidate boxes. These candidate boxes obtain corresponding bounding boxes through the network, resulting in a total of S $\times$ S $\times$ B bounding boxes. The terms 1_{ij}^{obj} and 1_{ij}^{noobj} indicate the presence or absence, respectively, of a target in the j^{th} detection box of the i^{th} grid. λ_{noobj} denotes the loss weight of the positioning error, C_i and $p_i(c)$ correspond to training values, and $\widehat{C}_i$ and $\widehat{p}_i(c)$ represent the prediction values.

Three model types are all developed based on the deep fine-tuning-based domain adaptation techniques but are 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 [58]. 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 seven 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, 750, and 1000, respectively. The second type of model was not notably 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 synthetic training sets with different ratios. The three synthetic training sets contained 1000 instances each, and the mixing ratios of target and source data were 1:9, 3:7, and 5:5, respectively. In total, fifteen models were developed: seven belonged to Type I, seven 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 Type I and Type II models. The second analysis case compared the performance of the Type II and III models. When comparing Type II and Type III, the Type II model was selected by matching the number of target instances with the Type III model. For example, when comparing a Type III model trained with a synthetic dataset of a 1 (100 instances) to 9 (900 instances) ratios, the selected Type II model should be trained with 100 training instances. The overall model performance was evaluated using four key metrics: precision, recall, F1 score, and mean average precision (mAP) [59–61]. 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 (7)–(10). Precision represents the fraction of truly classified objects to the total number of objects classified by the model as a given class. Recall indicates the ratio of the truly detected objects to the number of objects correctly predicted by the model [59,60].

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}} \tag{7}$$

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} \tag{8}$$

High percentages of both precision and recall indicate better model performance. The reason for this is that when the number of positive objects increases, the accuracy in 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 [61]; thus, a higher F1 score indicates better performance.

$$\text{F1 Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{9}$$

Then, all models were evaluated with mAP, calculated by averaging the average precision (A.P.) for all classes involved in the trained models. Higher mAP values reflect a well-performing model [59–61].

$$\text{mAP} = \frac{1}{n}\sum\nolimits_{k=1}^{k=n} AP_{\text{k}} \tag{10}$$

The computational cost of each model was also evaluated and compared by using the training time, which is intended to provide a more efficient method when benchmarking model development. In particular, the overall performance difference (11) between the models is evaluated by percentage variance [62], which can describe the degree of improvement and degradation in the new model's performance compared to the previous model.

$$\text{Percentage variance} = \frac{\text{New model type} - \text{Old model type}}{\text{Old model type}} \times 100 \tag{11}$$

After analyzing these models, the key findings of this study were extracted and summarized. Summarized vital findings would offer 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, the Type II model outperformed the other types of models, reducing the training time by more than two times in comparison to the other types. In the case of Model III, it worked well in both the target and source domains, although it consumed slightly more than the Type II model.

4. Benchmark

4.1. Dataset Acquisition

Four construction barrel targets from the United States, Korea, and Canada were selected for this study. The selected barrels are identical in terms of their purpose of use, as they are temporary control devices used on work roads. However, they comply with different standards in terms of shape, size, and layout under their traffic regulations. Such factors imply that a 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 a 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 4 k Ultra High Definition (UHD) 60 fps. Data were collected regardless of weather or time of day; therefore, the collected data varied in terms of weather conditions, lighting intensity, and shadows.

The collected data were then annotated with labeling rules to build the ground truth for this study. For instance, 15 consecutive frame-by-frame images were extracted from the collected data to provide variation in object angles and lighting while ensuring a reasonable balance between the amount of labeling data and the cost of labeling. Only barrels at a distance of 10–15 m from the data collection vehicle were labeled to ensure clear visibility and readability. These labeling rules were established with input from industry experts.

The constructed datasets for each domain were as follows: the Target S dataset consisted of 1000 images, the Target A dataset consisted of 657 images, Target B consisted of 657 images, and Target C consisted of 1000 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.

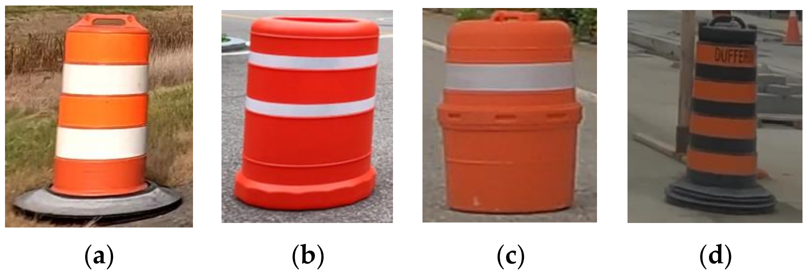


Figure 3. Example of barrel dataset: (**a**) Target S (U.S barrel), (**b**) Target A (Korean barrel for highway), (**c**) Target B (Korean barrel for short-term construction), and (**d**) Target C (Canadian barrel).

During data preprocessing, 1500 instances were randomly extracted from each target benchmark dataset for analysis and comparison in data preprocessing. The extracted instances were chosen to be as varied as possible, to reduce the potential bias of the data. After this, the extracted instances were split into training, testing, and validation sets with a ratio of 0.70, 0.15, and 0.15, respectively.

4.2. Model Development

Three types of models were developed 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. Type I models used open-sourced pre-trained weights from YOLOv5-L models trained on the COCO dataset, with the training set divided into seven subsets with varying instance amounts. Type II models used pre-trained weights from the source domain model instead of YOLOv5-L, while Type III models used source domain weights and synthetic training sets composed of randomly subsampled data from the source and target domains with different mixing ratios. For Types I and II, the training sets were further divided into seven subsets with varying instance amounts, while Type III used three synthetic datasets with a fixed total of 1000 instances each and different mixing ratios of target and source data. Table 1 provides an overview of the model types.

Table 1. Summary of model types.

Types		Model I	Model II	Model III
	Training Dataset	Only Target	Only Target	Mix (Source + Target)
	# Number of Training Sets (# Number of Instance)	7 (50, 100, 200, 300, 500, 750, and 1000)	7 (50, 100, 200, 300, 500, 750, and 1000)	3 mixing ratios (1:9, 3:7, and 5:5)
	Pre-Trained Model	YOLOv5L using COCO	Source Model	Source Model
Target	A (Korean Barrel for Highway)	7	7	3
	B (Korean barrel for short-term construction)	7	7	3
	C (Canada Barrel)	7	7	3
	Total constructed model	28	28	12

Each model type was trained, validated, and tested in the 11.11 Pytorch framework, an open-source machine learning tool 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 allows the modification of the weights of all layers of pre-trained networks when training the network for a new dataset [63]. The YOLOv5 model was improved by tuning two important learning parameters: learning rate and momentum. Since the YOLOv5 model utilizes an optimization algorithm that minimizes the loss of the regression problem while providing accurate estimates, these two hyperparameters are critical to achieving both high accuracy and fast processing speed [64]. In this study, each network was trained with a learning rate of 0.01 and a momentum of 0.937 using the stochastic gradient descent (SGD) optimization algorithm, as these values have been found to have the highest performance [65]. All training was run with 300 epochs with a batch size of 32 at an image input size of 640 $\times$ 640.

4.3. Performance and Cost Evaluation

Before developing the three types of models, this study tested existing source models on each of the target datasets to evaluate their performance in different national domains. Table 2 displays the overall testing results for each target testing set without training. The source models exhibited high performance metrics on the source datasets with a mAP of 0.985 and an F1 score of 0.958. However, their performance was poor on all performance metrics when applied directly to the target datasets without any training. This study focuses on detecting construction barrels for driver and construction safety, where correctly identifying the presence of a barrel is more important than correctly identifying its absence. However, when the source model was directly applied to the target dataset, its performance degraded the most, even though high recall is the most important aspect of the barrel detection model, confirming the difficulty of using the source model directly. Additionally, the lowest performance metric values were observed when the source model was applied to Target C, indicating that the model is inferior when the visual similarity between the source and target domains is low. These results demonstrate that domain differences can significantly impact model performance, particularly when detecting objects such as construction barrels. As a result, this study emphasizes the importance of considering domain differences when developing object detection models and highlights the necessity of implementing domain adaptation techniques.

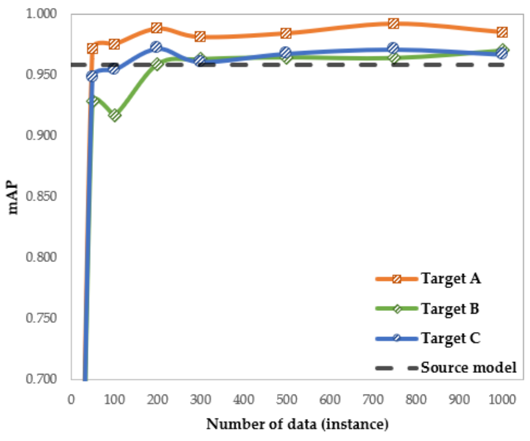
Table 2. Comparison of the testing results for each target dataset without training.

Testing Set	Precision	Recall	F1 Score	mAP_0.5 [1]
Source S	0.923	0.953	0.938	0.958
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

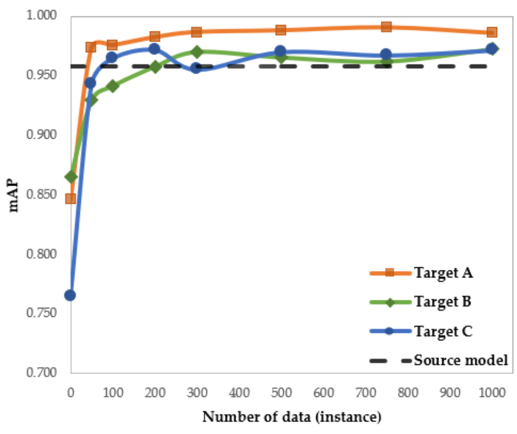
[1] Mean average precision at 0.5 Intersection over Union (IoU) threshold.

4.3.1. Analysis Case 1—Use of the Pre-Trained Models

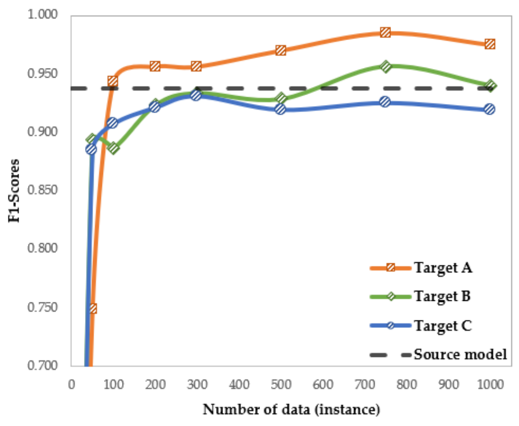
The performance of each model in target domains was compared for two models, Type I and Type II. In both types, the mAP values of all target models improved with the increase in labeled data in the training set. Among them, the Target A dataset outperformed the other target datasets in both Types I and II. Figure 4 illustrates the relationship between the model's performance and the amount of labeled data per target domain. In most aspects, 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 specific level of accuracy and even surpassed the original source model's performance (represented by the black dashed line in Figure 4).



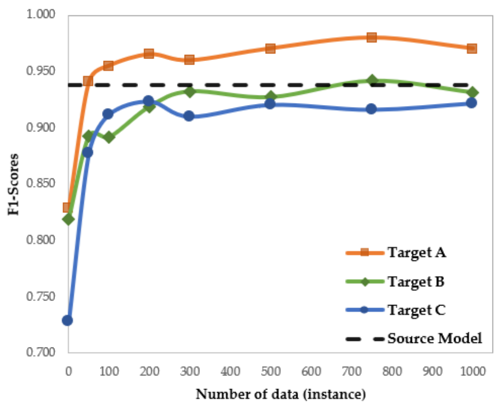
(**a**)



(**b**)



(**c**)



(**d**)

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, and (**d**) F1 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 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 the Type II model.

Table 3. Percentage variance of the computational cost of Model Types I and II.

Labeled Instances	Percentage Variance (%) [2] 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] The old model is Type I, and the new model is Type II.

4.3.2. Analysis Case 2—Required Target Data

In the context of both Model Type II and Model Type III, this study also evaluated the minimum amount of target data needed for effective domain adaptation by incorporating smaller datasets of varying sizes (10, 20, 30, 50, and 75). Furthermore, to ensure reliability, each dataset was trained three times using different randomly selected training data, while keeping the size of the training dataset constant. The dotted lines in Figures 5 and 6 represent the individual training datasets, while the solid lines represent the average of the three training datasets.

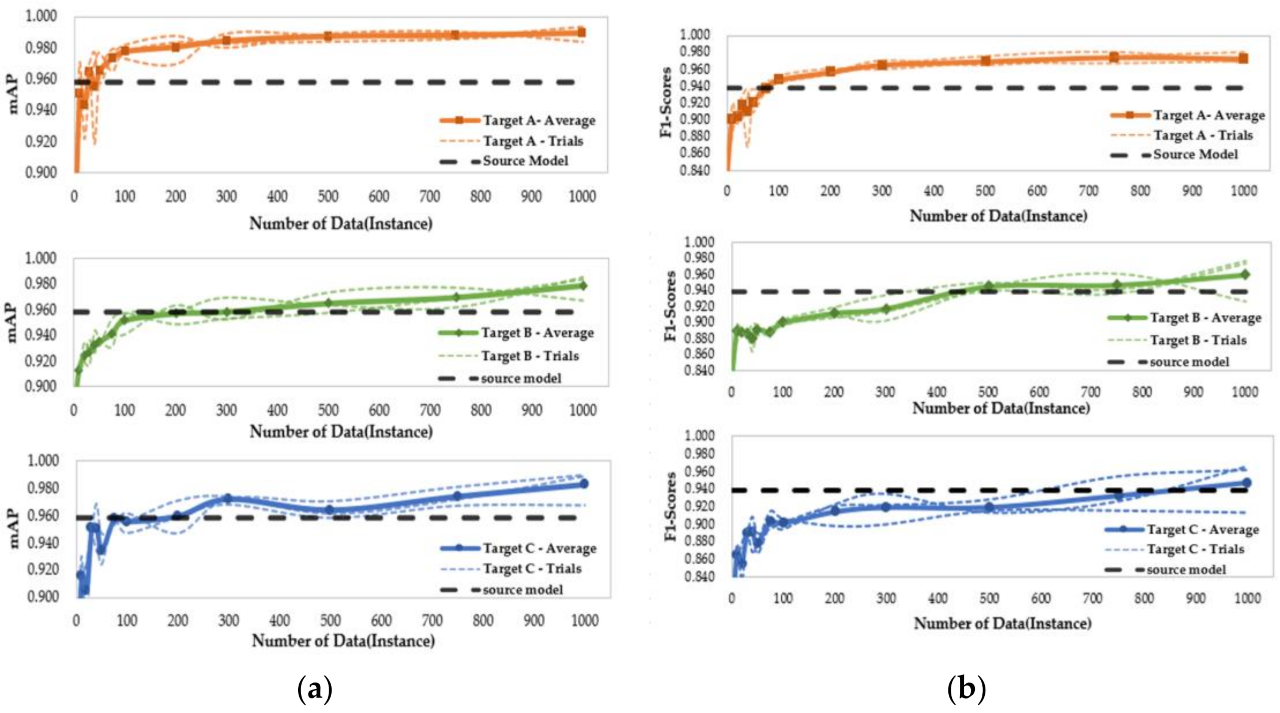
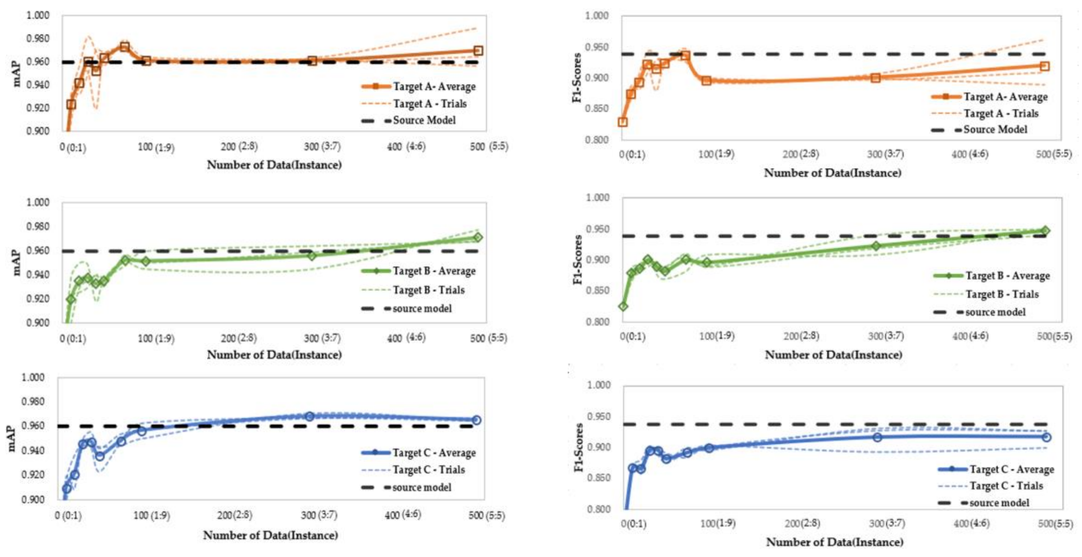


Figure 5. Three trials and the average of Model Type II. (**a**) mAP values under Model Type II; (**b**) F1 scores of Model Type II.



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

Figure 6. Three trials and the average of Model Type III. (**a**) mAP values of datasets with different mixing ratios; (**b**) F1 scores of datasets with different mixing ratios.

Focusing on Model Type II (see the Figure 5), although there were subtle differences between the target domains, similar trends were observed across all experiments. Specifically, for all three target domains, the mAP values increased up to 100 instances, after which they either increased slightly or remained constant. Different trends were observed up to 300 instances in each trial, but after 300 instances, the mAP values either increased slightly or remained constant with similar trends. Additionally, the mAP values were consistently higher than the original source model for all domains and trials above 500 instances. F1 was lower than the source model in most cases but increased slightly with higher target instance ratios.

For Model Type III (See the Figure 6), the mAP generally increased up to a mixture of 75 target domain instances and 925 source domain instances. When the mixed ratio of target instances exceeded 100 (1:9 ratio), the models for each target either converged at a certain level or experienced a slight increase. At a mixed ratio of approximately 3:7, the model's mAP values were higher than or equal to the performance of the source model for all targets, and at a ratio of 5:5, the mAP values surpassed the source model for all targets. Although F1 values were typically lower than those of the source model, they showed a slight increase as the target instance increased.

As expected, the Type II model required significantly less training time than the Type III model (as shown in Table 4). Both model types utilized the same amount of target data; however, the actual volume of training data differed due to the Type III model having a fixed number of instances (1000). Specifically, when the data ratio between the source and target domains was highly imbalanced, the difference in training time between the two model types was considerable. Conversely, when the ratio was more evenly distributed (5:5), the disparity in training time between the two model types appeared to decrease compared to scenarios with more uneven ratios.

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**
(1:9) 100:900	495%	1560%	849%
(3:7) 300:700	230%	241%	314%
(5:5) 500:500	17%	138%	253%

[3] The 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 are available. According to the untrained test results, the model performance was 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 around 18% in the Target C domain, which had the lowest visual similarity to the source domain. This implies that the Target C domain had 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 was degraded in 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. In addition, 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 7.

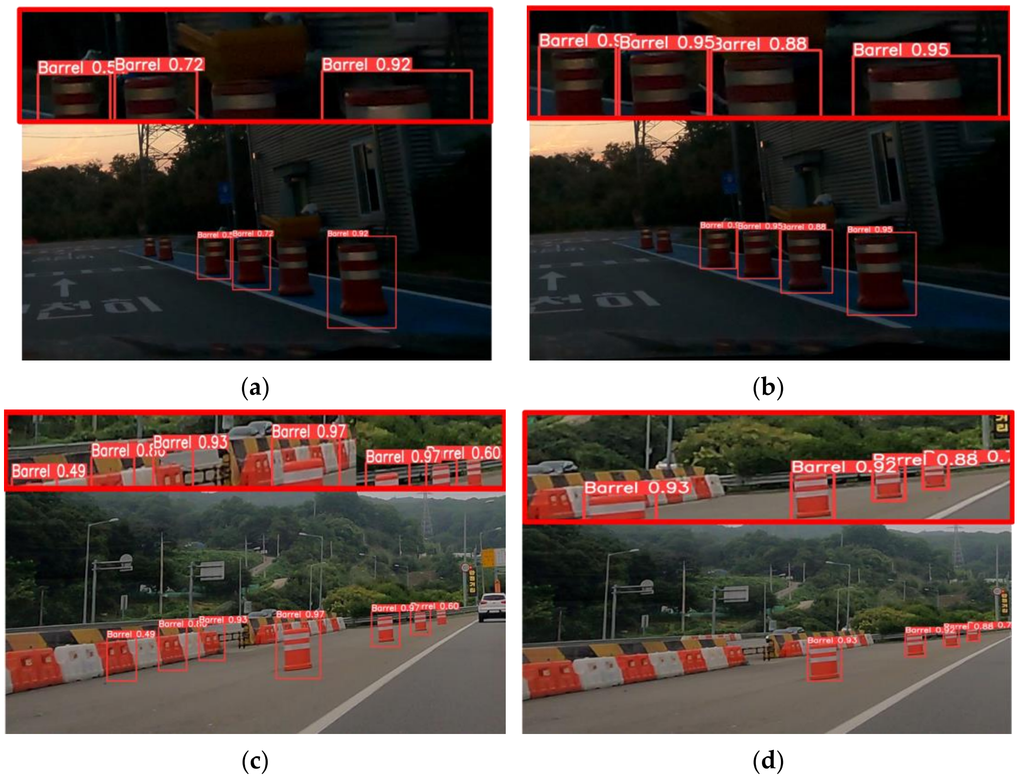


Figure 7. Example of detection results. (**a**) Fewer T.P. results of Model Type I; (**b**) more T.P. results of Model Type II; (**c**) more T.N. results of Model Type I; and (**d**) fewer T.N. results of Model Type II.

This study also showed that the performance of all types of models improved significantly, even when trained with a small amount of target data. Among the different models tested, Model II exhibited the best adaptation performance with the lowest 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 showing that reusing knowledge of object features already learned from other models helps to identify the best weights quickly by sharing similar features [66].

However, it was observed that Model II's performance degraded significantly when applied to the source domain, as displayed in Table 5. This occurrence is known as "catastrophic forgetting". In the context of domain adaptation, catastrophic forgetting refers to the phenomenon that well-fine-tuned models in the target domain often degrade in performance when re-applied to the source domain [67]. Thus, these results showed that Model II was only suitable for the target domain. Model III, on the other hand, took longer to train due to the larger amount of training data required but performed well in both source and target domains. The mAP of the model decreased by a maximum of 2%, which was considered negligible. Thus, Model III appears to be interchangeable between source and target domains with less catastrophic forgetting.

Table 5. Comparisons of Model Types 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	(1:9) 100	0.954	0.963	0.961	0%	1%	0%
	(3:7) 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 when increasing the number of labeled data in the training set. This trend has been found in other studies in this field [68,69]. 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. Moreover, 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 Figures 4–6), most models' performance sharply increased until it reached 200 instances. Moreover, when comparing the mAP performance of the models with the number of instances over multiple trials, the values of each trial were different until 300 instances, after which the performance of the models gradually converged or remained constant. Additionally, for all model cases, when the number of training instances was greater than 500, the mAP values were higher than the mAP values of the original source model.

To summarize, the most cost-effective way to build a model in the new domain was to use Model Type II with a number of training instances of about 300 to 500 (around 3–5% of the source data). However, the domain adaptation method of fine-tuning-based transfer learning used in this study showed a phenomenon of catastrophic forgetting, where the model performs well in the new domain, but the performance decreases somewhat when it is reapplied to the source domain. On the other hand, when Model Type III is used to build a model in a new domain, it is not as efficient as Model Type II in terms of training time and performance, but catastrophic forgetting rarely occurs.

6. Conclusions

Object detection using computer vision technology has the potential to significantly reduce the practitioner's workload through timely and frequent inspections in the field of TCD management. It is worth noting that the majority of training datasets for such models are found in highly advanced AI countries, while countries with less developed AI capabilities, such as South Korea, collaborate with these nations to address data scarcity [70]. In light of this, the current study offers insights into the optimal amount of data necessary for cost-effective model training in order to reach a specific accuracy level when developing a new object detection model using imported data from models created in other countries.

During the model development process, this study utilized a domain adaptation technique that leverages pre-trained models from related domains to generate three model cases with different pre-trained weights and training sets. After comparing and analyzing the three model cases, it was determined that the source domain model cannot be directly applied to a new target domain, necessitating even a small amount of domain-specific data for optimal model performance. The most cost-effective approach to building the model was found to be the fine-tuning-based transfer learning technique (Model Type II), which uses around 300–500 training data (about 3–5% of the source model data) to train the model with pre-trained weights in the target domain. However, the newly trained model in the target domain exhibited a performance degradation known as catastrophic forgetting when tested on the source domain. When training with mixed source and target data (Model Type III), the model could prevent the catastrophic forgetting issue, but Model Type II remained the most cost-effective option. These findings indicate that additional research, such as continuous learning, incremental training, and lifelong learning, is necessary to prevent catastrophic forgetting while training models in multiple domains without source data.

While this study can serve as a useful benchmark for related fields, it still has several limitations. First, the benchmark did not take into account different categories of TCD objects, second, it was designed for practitioners working with limited data and therefore did not take into account different dataset sizes, and finally, it did not take into account the use of multiple source domains. As an initial investigation in this area, we have plans to improve the developed model by applying statistical methodologies such as bootstrapping to improve sample quantification and produce more statistically robust results. This approach will allow future research to reduce reliance on heuristics and increase the number of experiments, ultimately leading to more meaningful results. The developed model will also be compared quantitatively and qualitatively to TCD models using recently published algorithms.

Author Contributions: Conceptualization, D.O., S.S. and K.K. (Kyubyung Kang); methodology, D.O. and K.K. (Kyubyung Kang); software, D.O., J.X. and K.J.; validation, D.O.; formal analysis, D.O. and K.K. (Kyubyung Kang); investigation, D.O. and S.S.; resources, D.O., S.S. and K.K. (Kyubyung Kang); data curation, D.O., S.S., K.J. and J.X.; writing—original draft preparation, D.O.; writing—review and editing, K.K. (Kibum Kim) and K.K. (Kyubyung Kang); visualization, D.O.; supervision, K.K. (Kyubyung Kang) and H.P.; project administration, H.P. and J.W.; funding acquisition, H.P. and J.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data generated or analyzed during this study are available from the corresponding author by request.

Acknowledgments: This research was supported by the Chungbuk National University BK21 Program (2022).

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. [CrossRef]
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. [CrossRef]

3. Chian, E.; Fang, W.; Goh, Y.; Tian, J. Computer vision approaches for detecting missing barricades. *Autom. Constr.* **2021**, *131*, 103862. [CrossRef]
4. Yan, X.; Zhang, H. Computer vision-based disruption management for prefabricated building construction schedule. *J. Comput. Civ. Eng.* **2021**, *35*, 04021027. [CrossRef]
5. Paneru, S.; Jeelani, I. Computer vision applications in construction: Current state, opportunities & challenges. *Autom. Constr.* **2021**, *132*, 103940. [CrossRef]
6. Zaidi, S.; Ansari, M.; Aslam, A.; Kanwal, N.; Asghar, M.; Lee, B. A survey of modern deep learning-based object detection models. *Digit. Signal Process.* **2022**, *126*, 103514. [CrossRef]
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. [CrossRef]
8. Zhu, X.; Vondrick, C.; Ramanan, D.; Fowlkes, C.C. Do We Need More Training Data or Better Models for Object Detection. *BMVC* **2012**, *3*, 11.
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. In *IbPRIA*; Springer: Cham, Switzerland, 2022; pp. 629–641. [CrossRef]
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*, 1380–1390. [CrossRef]
11. Zhuang, F.; Qi, Z.; Duan, K.; Xi, D.; Zhu, Y.; Zhu, H.; Xiong, H.; He, Q. A comprehensive survey on transfer learning. *Proc. IEEE* **2020**, *109*, 43–76. [CrossRef]
12. Wang, M.; Deng, W. Deep visual domain adaptation: A survey. *Neurocomputing* **2018**, *312*, 135–153. [CrossRef]
13. Hsu, H.; Yao, C.; Tsai, Y.; Hung, W.; Tseng, H.; Singh, M.; Yang, M. Progressive Domain Adaptation for Object Detection. In Proceedings of the 2020 IEEE Winter Conference on Applications of Computer Vision (WACV), Snowmass, CO, USA, 1–5 March 2020; pp. 738–746. [CrossRef]
14. Zhao, S.; Yue, X.; Zhang, S.; Li, B.; Zhao, H.; Wu, B.; Krishna, R.; Gonzalez, J.E.; Sangiovanni-Vincentelli, A.L.; Seshia, S.A.; et al. A review of single-source deep unsupervised visual domain adaptation. *IEEE Trans. Neural Netw. Learn. Syst.* **2020**, *33*, 473–493. [CrossRef]
15. Tseng, H.; Lee, H.; Huang, J.; Yang, M. Cross-domain few-shot classification via learned feature-wise transformation. *arXiv* **2020**, arXiv:2001.08735. [CrossRef]
16. Kim, Y.; Cho, D.; Han, K.; Panda, P.; Hong, S. Domain adaptation without source data. *IEEE TAI* **2021**, *2*, 508–518. [CrossRef]
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*, 175. [CrossRef]
18. Wang, Z.; Yang, J.; Jiang, H.; Fan, X. CNN training with twenty samples for crack detection via data augmentation. *Sensors* **2020**, *20*, 4849. [CrossRef] [PubMed]
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 Sens.* **2021**, *13*, 4386. [CrossRef]
20. Stallkamp, J.; Schlipsing, M.; Salmen, J.; Igel, C. The German Traffic Sign Recognition Benchmark: A multi-class classification competition. In Proceedings of the 2011 International Joint Conference on Neural Networks, San Jose, CA, USA, 31 July–5 August 2011; pp. 1453–1460. [CrossRef]
21. Timofte, R.; Zimmermann, K.; Van Gool, L. Multi-view traffic sign detection, recognition, and 3D localisation. *Mach. Vis. Appl.* **2014**, *25*, 633–647. [CrossRef]
22. Larsson, F.; Felsberg, M. Using Fourier Descriptors and Spatial Models for Traffic Sign Recognition. In *Image Analysis SCIA*, 2nd ed.; Heyden, A., Kahl, F., Eds.; Springer: Berlin/Heidelberg, Germany, 2011; Volume 6688, pp. 238–249. [CrossRef]
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*, 1484–1497. [CrossRef]
24. Zhu, Z.; Liang, D.; Zhang, S.; Huang, X.; Li, B.; Hu, S. Traffic-Sign Detection and Classification in the Wild. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 2110–2118. [CrossRef]
25. Dhall, A.; Dai, D.; Van Gool, L. Real-time 3D Traffic Cone Detection for Autonomous Driving. In Proceedings of the 2019 IEEE Intelligent Vehicles Symposium (IV), Paris, France, 9–12 June 2019; pp. 494–501. [CrossRef]
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*, 2nd ed.; Krouska, A., Troussas, C., Caro, J., Eds.; Springer: Cham, Switzerland, 2022; Volume 556, pp. 382–391. [CrossRef]
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 Sens.* **2020**, *12*, 3837. [CrossRef]
28. Kim, Y.; Song, K.; Kang, K. *Framework for Machine Learning-Based Pavement Marking Inspection and Geohash-Based Monitoring*; ICTD: Seattle, WA, USA, 2022; pp. 123–132. [CrossRef]
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. *Constr. Res. Congr.* **2022**, 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: Chicago, IL, USA, 2021; pp. 15–18.

31. Csurka, G. Domain adaptation for visual applications: A comprehensive survey. *arXiv* **2017**, arXiv:1702.05374.
32. Yosinski, J.; Clune, J.; Bengio, Y.; Lipson, H. How transferable are features in deep neural networks? *arXiv* **2014**, arXiv:1411.1792.
33. Shahinfar, S.; Meek, P.; 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. *Ecol. Inform.* **2020**, *57*, 101085. [CrossRef]
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*, 563. [CrossRef]
35. Guo, Y.; Shi, H.; Kumar, H.; Grauman, K.; Rosing, T.; Feris, R. SpotTune: Transfer Learning Through Adaptive Fine-Tuning. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019; pp. 4800–4809. [CrossRef]
36. Misra, I.; Shrivastava, A.; Gupta, A.; Hebert, M. Cross-Stitch Networks for Multi-task Learning. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 3994–4003. [CrossRef]
37. Bengio, Y. Deep learning of representations for unsupervised and transfer learning. In Proceedings of the ICML Workshop on Unsupervised and Transfer Learning. JMLR Workshop and Conference Proceedings, Washington, DC, USA, 2 July 2011; pp. 17–36.
38. Bruzzone, L.; Marconcini, M. Domain adaptation problems: A dasvm classification technique and a circular validation strategy. *IEEE TPAMI* **2010**, *32*, 770–787. [CrossRef] [PubMed]
39. Chu, W.S.; De la Torre, F.; Cohn, J.F. Selective transfer machine for personalized facial action unit detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, 23–28 June 2013; pp. 3515–3522.
40. Gong, B.; Grauman, K.; Sha, F. Connecting the dots with landmarks: Discriminatively learning domain-invariant features for unsupervised domain adaptation. *PMLR* **2013**, *28*, 222–230.
41. Pan, S.J.; Yang, Q. A survey on transfer learning. *IEEE Trans. Knowl. Data Eng.* **2010**, *22*, 1345–1359. [CrossRef]
42. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative adversarial nets. *NIPS* **2014**, *63*, 2672–2680. [CrossRef]
43. Xu, W.; He, J.; Shu, Y. Transfer learning and deep domain adaptation. *Adv. Appl. Deep Learn.* **2020**, *45*. [CrossRef]
44. Rostami, M.; Kolouri, S.; Eaton, E.; Kim, K. Deep transfer learning for few-shot SAR image classification. *Remote Sens.* **2019**, *11*, 1374. [CrossRef]
45. Wang, W.; Ma, L.; Chen, M.; Du, Q. Joint correlation alignment-based graph neural network for domain adaptation of multitemporal hyperspectral remote sensing images. *IEEE J-STARS* **2021**, *14*, 3170–3184. [CrossRef]
46. Lasloum, T.; Alhichri, H.; Bazi, Y.; Alajlan, N. SSDAN: Multi-source semi-supervised domain adaptation network for remote sensing scene classification. *Remote Sens.* **2021**, *13*, 3861. [CrossRef]
47. Zheng, Z.; Zhong, Y.; Su, Y.; Ma, A. Domain adaptation via a task-specific classifier framework for remote sensing cross-scene classification. *IEEE Trans Geosci. Remote Sens.* **2022**, *60*, 4416212. [CrossRef]
48. Federal Highway Administration (FHWA). *Manual on Uniform Traffic Control Devices (MUTCD)*; FHWA: Washington, DC, USA, 2009.
49. Ministry of Land, Infrastructure and Transportation (MOLIT). *Traffic Management Guidelines for Road 25 Construction Sites*; MOLIT: Sejong, Republic of Korea, 2018.
50. Ontario Traffic Manual (OTM). *Book 7: Temporary Conditions*; Ontario Ministry of Transportation: St. Catherines, ON, Canada, 2014.
51. Jocher, G.; Chaurasia, A.; Stoken, A.; Borovec, J.; Kwon, Y.; Michael, K. ultralytics/yolov5: v7. 0-YOLOv5 SOTA Realtime Instance Segmentation; 2022. Zenodo. Available online: https://zenodo.org/record/7347926#.ZGHTqXxByUk (accessed on 16 April 2023).
52. Wang, J.; Chen, Y.; Dong, Z.; Gao, M. Improved YOLOv5 network for real-time multi-scale traffic sign detection. *Neural Comput. Appl.* **2022**, *35*, 7853–7865. [CrossRef]
53. Zhu, Y.; Yan, W.Q. Traffic sign recognition based on deep learning. *Multimed. Tools Appl.* **2022**, *81*, 17779–17791. [CrossRef]
54. He, K.; Zhang, X.; Ren, S.; Sun, J. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. *IEEE TPAMI* **2015**, *37*, 1904–1916. [CrossRef]
55. Golubeva, A.; Neyshabur, B.; Gur-Ari, G. Are wider nets better given the same number of parameters? *arXiv* **2020**, arXiv:2010.14495.
56. 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. [CrossRef]
57. Zheng, Z.; Wang, P.; Liu, W.; Li, J.; Ye, R.; Ren, D. Distance-IoU Loss: Faster and Better Learning for Bounding Box Regression. In Proceedings of the 2020 AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; pp. 12993–13000. [CrossRef]
58. Lin, T.Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C.L. Microsoft COCO: Common Objects in Context. In *Lecture Notes in Computer Science*, 2nd ed.; Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T., Eds.; Springer: Cham, Switzerland, 2014; Volume 8693. [CrossRef]
59. Padilla, R.; Passos, W.L.; Dias, T.L.B.; Netto, S.L.; da Silva, E.A. A Survey on Performance Metrics for Object-Detection Algorithms. In Proceedings of the 2020 International Conference on Systems, Signals and Image Processing (IWSSIP), Niteroi, Brazil, 1–3 July 2020; pp. 237–242. [CrossRef]

60. 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*, 142–149. [CrossRef]
61. 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. [CrossRef]
62. Brechner, R.; Bergeman, G. *Contemporary Mathematics for Business & Consumers*, 8th ed.; Cengage Learning: Boston, MA, USA, 2016.
63. Tajbakhsh, N.N.; Shin, J.; Gurudu, S.; Hurst, R.; Kendall, C.; Gotway, M.; Liang, J. Convolutional Neural Networks for Medical Image Analysis: Full Training or Fine Tuning? *IEEE Trans. Med. Imaging* **2016**, *35*, 1299–1312. [CrossRef]
64. Smith, L.N. A disciplined approach to neural network hyper-parameters: Part 1—Learning rate, batch size, momentum, and weight decay. *arXiv* **2018**, arXiv:1803.09820.
65. Isa, I.S.; Rosli, M.S.A.; Yusof, U.K.; Maruzuki, M.I.F.; Sulaiman, S.N. Optimizing the Hyperparameter Tuning of YOLOv5 for Underwater Detection. *IEEE Access* **2022**, *10*, 52818–52831. [CrossRef]
66. 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, Y.D., Bhateja, V., Majhi, R., Eds.; Springer: Cham, Switzerland, 2021; Volume 1177, pp. 359–367. [CrossRef]
67. Xu, Y.; Zhong, X.; Yepes, A.J.J.; Lau, J.H. Forget me not: Reducing catastrophic forgetting for domain adaptation in reading comprehension. In Proceedings of the 2020 International Joint Conference on Neural Networks (IJCNN), Glasgow, UK, 19–24 July 2020; pp. 1–8. [CrossRef]
68. Sun, C.; Shrivastava, A.; Singh, S.; Gupta, A. Revisiting Unreasonable Effectiveness of Data in Deep Learning Era. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 843–852. [CrossRef]
69. Tabak, M.A.; Norouzzadeh, M.S.; Wolfson, D.W.; Sweeney, S.J.; VerCauteren, K.C.; Snow, N.P.; Halseth, J.M.; Di Salvo, P.A.; Lewis, J.S.; White, M.D.; et al. Machine learning to classify animal species in camera trap images: Applications in ecology. *Methods Ecol. Evol.* **2019**, *10*, 585–590. [CrossRef]
70. Korea Ministry of Science and ICT. *National Strategy for Artificial Intelligence*; MIST: Sejong-Si, Republic of Korea, 2019.