Road Damage Detection and Classification with Detectron2 and Faster R-CNN

Vung Pham
Computer Science Department
Texas Tech University
Lubbock, USA
vung.pham@ttu.edu

Chau Pham
Computer Science Department
Texas Tech University
Lubbock, USA
chaupham@ttu.edu

Tommy Dang
Computer Science Department
Texas Tech University
Lubbock, USA
tommy.dang@ttu.edu

2020 IEEE International Conference on Big Data (Big Data) | 978-1-7281-6251-5/20/$31.00 ©2020 IEEE | DOI: 10.1109/BigData50022.2020.9378027

***Abstract*—The road is vital for many aspects of life, and road maintenance is crucial for human safety. One of the critical tasks to allow timely repair of road damages is to quickly and efficiently detect and classify them. This work details the strategies and experiments evaluated for these tasks. Specifically, we evaluate Detectron2's implementation of Faster R-CNN using different base models and configurations. We also experiment with these approaches using the Global Road Damage Detection Challenge 2020, A Track in the IEEE Big Data 2020 Big Data Cup Challenge dataset. The results show that the X101-FPN base model for Faster R-CNN with Detectron2's default configurations is efficient and general enough to be transferable to different countries in this challenge. This approach results in F1 scores of 51.0% and 51.4% for the test1 and test2 sets of the challenge, respectively. Though the visualizations show good prediction results, the F1 scores are low. Therefore, we also evaluate the prediction results against the existing annotations and discover some discrepancies. Thus, we also suggest strategies to improve the labeling process for this dataset.**

***Index Terms*—Object detection, Detectron2, Road damage detection, Faster R-CNN and classification, Transferable learning**

I. Introduction

The road is crucial in different aspects of life, from economic development and social benefits to safety. Therefore, road maintenance is vital for all countries in the world. One of the road maintenance tasks is to accurately detect damages and then devote efficient repairs in a timely manner. However, for most countries, road crack detection and classification are currently based on human manual works or expensive sensors. Therefore, automatic detection and classification of road damage types are getting popular recently. Also, deep learning, with its recent advancements, is gaining traction and has state-of-the-art results in various computer vision tasks [1]. Therefore, many works in the literature use deep learning approaches to detect and classify road damages.

Using deep learning for road damage detection and classification often involves three sub-tasks: 1) collecting image data, 2) creating labels for the data, and 3) building deep learning models from the labeled data. While collecting data can be done efficiently using mobile devices with GPS and camera [2], the labeling process takes time, and the detection/classification results are still limited. Also, the models learned from the data coming from one country often are not generalized enough to be transferable to different countries [3]. Additionally, providing bounding boxes and labels for road damages is error-prone and demands a massive amount of human labor to have accurate results.

Therefore, this work explores state-of-the-art object detection methods to find general road damage detection and classification model that is usable for different territories. We then apply the selected approaches to the Global Road Damage Detection Challenge 2020, A Track in the IEEE Big Data 2020 Big Data Cup Challenge [4] dataset. Furthermore, we also evaluate the quality of the currently labeled data and propose a strategy to generate more labeled data. Thus, our contributions are:

- Exploring current state-of-the-art object detection methods and their applicability to road damage detection and classification tasks.
- Experimenting with these approaches using the Global Road Damage Detection Challenge 2020 dataset to find one single model that is efficient and can be transferable to different territories.
- Visualizing the prediction results and qualitatively evaluating the existing annotations follows by giving suggestions to improve the labeling process for this dataset.

II. Dataset and evaluation method

The dataset used in this work is from the Global Road Damage Detection Challenge 2020 [4]. This dataset consists of one train set (*train*) and two test sets (*test1* and *test2*). Specifically, the training and testing sets contain road damage images, bounding boxes, and damage types (for training set) from three countries (i.e., Czech, India, and Japan). Furthermore, for this dataset, there are four types of labelled road damage types namely Longitudinal Crack, Transverse Crack, Aligator Crack, and Pothole (labelled *D00*, *D10*, *D20*, and *D40*, respectively). For this section's brevity, we defer further details about the number of images and damage type distributions in the individual countries and the whole dataset to Section IV-A.

For this challenge, the evaluation method is based on the *F1* score defined to balance the precision (p) and recall (r).

These are defined as:

$$p = \frac{C_d}{P_d},\ r = \frac{C_d}{A_d},\ F1 = \frac{2pr}{p+r}$$

where C_d, P_d, and A_d are the numbers of correctly predicted damages, the predicted damages, and all the ground-truth damages from the evaluating set, respectively. Furthermore, the definition of correctly predicted damage has two criteria. They are 1) the predicted bounding box must match, and 2) the predicted label is correct. The latter is obvious, and the former is defined by the Intersection over Union (*IoU*) score, which is defined as follows:

$$IoU = \frac{area(P_b \cap G_b)}{area(P_b \cup G_b)}$$

where P_b and G_b are a predicted box and a ground-truth box, respectively. Also, $area(P_b \cap G_b)$ and $area(P_b \cup G_b)$ means the areas of the intersection and the union between the two boxes, correspondingly. In this case, if $IoU >= 0.5$, then it is a match, and it is not otherwise.

III. Related work

Object detection using deep neural networks is an emerging field, and it is not the purpose of this work to review all the recent results in this field. Also, object detection techniques are continually evolving, and comparisons might be outdated quickly. Instead, we briefly survey families of current state-of-the-art object detection methods in general and methods related to road damage detection and classification specifically.

A. Deep learning based object detection

Deep learning-based object detection is gaining initial success, and there are many works in the literature regarding this. We refer interested readers to [5] for a good survey of these methods. Recent techniques are generally related to two prominent families, namely Region-Based Convolutional Neural Networks (R-CNNs) and You Only Look Once (YOLO).

Ross Girshick et al. [6] propose R-CNNs approach with three main modules. The first one is a region proposal module that generates candidate regions (bounding boxes) using computer vision techniques. The second one is the feature extraction module. This second module uses convolutional neural networks to extract the features from the candidate regions. Finally, the last module is a classifier that predicts the classes of the proposed candidates using the extracted features.

R-CNNs takes a long time to train because training is done in multiple stages. Besides training, the prediction stage is also slow. Therefore, Girshick proposes another model called Fast R-CNN [7] to tackle these issues. Fast R-CNN is trained as a single model instead of three separate modules. This architecture takes the images and proposes candidate regions, then passes them through a popular, pre-trained image classification model (e.g., ResNet [8], VGG-16 [9]) to extract features from the candidates. The extracted features then undergo a Region of Interest (RoI) pooling layer, followed by two fully connected layers. Finally, there are two other fully connected heads for bounding box regression and label classification purposes.

Though Fast R-CNN improves the training and predicting time, it still needs the region proposal as the inputs. In other words, the region proposals for each image still needs to be done separately (e.g., using image processing techniques). Therefore, Ren et al., [10] propose Faster R-CNN to tackle this issue. Its main improvement is the ability to incorporate region proposals as a part of the final model using Region Proposal Network (RPN). In other words, there are two smaller networks in this architecture. The first one is a Region Proposal Network (RPN), and the second one is the Fast R-CNN. These two sub-networks are trained simultaneously, though for two different tasks: 1) region proposals and 2) bounding box classification and regression. These strategies help to improve the training and object detection time and accuracy.

Another famous family of object detection is YOLO with different versions such as YOLO [11], YOLOv2 [12], YOLOv3 [13], and YOLOv4 [14]. Different YOLO versions may differ in terms of architectures and techniques used. However, generally, it involves a single neural network, with the input being the images and ground-truth boxes/segments and labels (while training). The outputs are the bounding boxes and corresponding labels of the detected objects from the image. Specifically, it divides an image into a grid of cells. Feature extracted from each cell is used to predict objects with centers of the bounding boxes that fall into the cell. The advantage of this method is that it is faster to train and predict. However, the benefit comes with a slightly lower accuracy compared to Faster R-CNNs.

While YOLO mainly has its advantage for speed and Faster R-CNN is better at accuracy, Single Short Detection (SSD) [15] allows a better balance between speed and accuracy. SSD runs a convolutional neural network on input image only one time and computes a feature map. It also uses anchor boxes of different sizes and aspects as Faster R-CNN. Regarding bounding box sizes, SSD predicts them at different convolutional layers. The reason is that convolutional layers have different receptive fields of the inputs. In other words, the deeper a convolutional layer is, the larger its receptive field will be. Thus, the deeper convolutional layer features are used to predict larger bounding boxes and vice versa.

B. Deep learning based road damage detection

Deep learning is gaining success in various areas such as efficient manufacturing and system operations [16], estimating visual features [17], and solar flare event predictions [], to name but a few. Road damage detection and classification is no exception. Various works in the literature use deep learning for these tasks. With the help of GPS- and camera-enabled mobile devices, it is now relatively easy to collect road images for this purpose. For instance, Maeda et al., [2] propose using a smartphone placed on a car's dashboard to collect pictures of road cracks, label them, and make them available for the public. Based on this dataset, Yanbo Wang et al., [18] propose to use Faster R-CNN and SSD with

pre-trained ResNet and VGG as bases to tackle the damage detection and classification tasks. Also, they aggregate up to 14 Faster R-CNN models and 2 SSD models to improve the detection result. Similarly, Wenzhe Wang et al., [19] also propose to use Faster R-CNN with data exploration step to adjust appropriate hyperparameters such as anchor boxes and ratios. Furthermore, they use different augmentation types (such as contrast transformation, brightness adjustment, and Gaussian blur) to improve their results.

On the other hand, [20] uses YOLOv3 with *darknet53* as the base model to tackle these tasks. They also experiment with two augmentation strategies. The first strategy is to generate more images for damage types with lower number occurrences using brightening or gray-scale, and the second strategy is to use cropping. However, in general, augmentations do not help. Additionally, Kluger et al., [21] utilize Faster R-CNN, RetinaNet [22], and Convolutional Neural Network combined with Random Forest to solve the tasks. The use of Random Forest is due to the assumption that it helps in case there are few samples in the training data [23]. Furthermore, they propose (without validating the impacts of the proposal) to use Cycle-Consistent Adversarial Networks (CycleGAN) [24] for data augmentation. Their experiments also show that Faster R-CNN works best for road damage classification and detection.

In a recent study, Maeda et al., [25] use Generative Adversarial Nets (GAN) [26] to generate damages with a lower number of occurrences to improve the detection results for this specific type of damages. Specifically, they use Progressive Growing GAN (PG-GAN) [27] to artificially generate more damages of *pothole* damage type and improve prediction results for this type of cracks in Japan. The reason is that Japan has a low number of *pothole* damages. Furthermore, they also use Poisson blending [28] to place the generated damages to the existing images to make the artificial patches look more natural to its containers. In another work, Arya et al., [3] suggest that the current techniques are not transferable from one country to another. Therefore, such a model is needed to save data collection, data labeling, and training time.

All in all, these works show that Faster R-CNN seems to be a useful technique for road damage detection and classification with a good trade-off between time and accuracy. These works also indicate that except when we only focus on a damage type with a small number of occurrences (i.e., *pothole* damages in Japan), data augmentations (other than the obvious ones such as horizontal flipping and resizing) do not generally help. Furthermore, several of these works use ensembles to improve their prediction results. Though ensembles help improve the results, they also significantly increase training and prediction time and are not encouraged in this year's challenge.

IV. Methodology

Our general methodology is that we start with the data exploration stage to understand the dataset. We then proceed by splitting the training dataset further into the training and evaluation sets. The validation enables us to evaluate the hyper-parameters for our architectures quantitatively. Regarding deep learning model architectures, we start with the commonly used model architectures for road damage detection and classification tasks. We then proceed with strategies to improve the base models, such as changing hyper-parameters, train data augmentations, and test time augmentations. It's worth noting that ensembles and individual models for individual countries would have better prediction results. However, it is restricted by the challenge that we should have a single-algorithm and single-model approach. Therefore, we do not attempt these directions.

TABLE I
Number of training iterations, test score thresholds, and F1 Scores for different experiments.

	R101	**X101**	**X101+Augs**	**X101+SRs**
Converge iteration	85,000	105,000	135,000	70,000
Score threshold	0.65	0.71	0.51	0.71
F1 Score	0.5285	0.5425	0.5398	0.5451

R101: Model with R101 as the base model
X101: Model with X101 as the base model
X101+Augs: Model with X101 as the base and uses augmentations
X101+SRs: Model with X101 as the base and uses custom sizes and ratios
Converge iteration: The iteration at which the model provides best result
Score threshold: Test score threshold
F1 Score: F1 score of the predicted results on evaluation set

Table I summarizes the main experiments that we have made with their corresponding training iterations (i.e., converge iterations at which it produces the best results on evaluation data), test score threshold (i.e., the threshold used to decide if road damage exists), and corresponding F1 Scores on the evaluation dataset. The following sections detail data exploration with train/evaluation splits and these experimented models.

A. Data exploration and train/evaluation splits

This dataset consists of one training set (*train*) and two test sets (*test1* and *test2*). The training set contains 21,041 images (2,829, 7,706, and 10,506 for Czech, India, and Japan, respectively). The two test sets contain 2,631 and 2,664 images, correspondingly. There are 34,702 ground-truth labels (bounding boxes and damage types) in the training set. Specifically, Figure 1 shows the damage type distributions (of the four corresponding types) over the three countries. In general, Japan has higher numbers of images and damages, and the pothole damage type has the lowest number of occurrences.

On the other hand, India has fewer images and damage labels. Also, *D04* is the one with the highest number of occurrences, while there are only a few *D01* damages in India. Finally, Czech has the fewest number of images and damage samples (a total of 1,745 labels). It's worth noting that due to different numbers of images, different damage types distributions over different countries, it is difficult to have a transferable model for all three countries. In other words, having different models for different countries (with a sufficient number of images like Japan and India) should have better accuracy compared to a single model for all three countries.

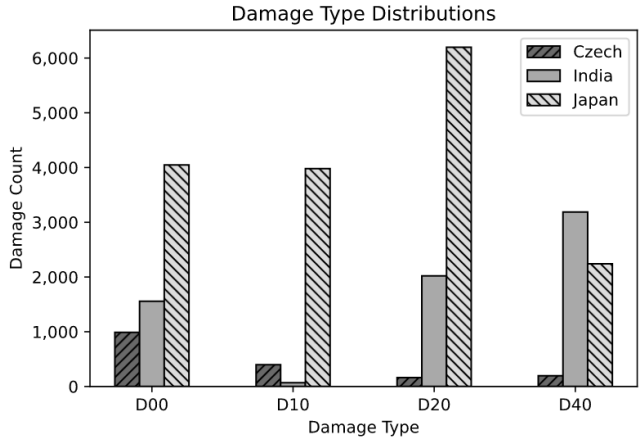

Fig. 1. Damage type distributions over three countries (Czech, India, and Japan). It's observable that different countries have a different number of damages for the four types.

We also split the training dataset into training and validation sets with the stratify as the origin of the damage types (i.e., country). Specifically, we keep 90% of the images for training and 10% for evaluation. This 10% for the evaluation split is reasonable because it results in 1,221 images in this set. More than a thousand images are general enough to evaluate the performance of the learned models. This evaluation set is used to quantitatively evaluate our model's performance and the hyperparameter selection process during training (such as prediction score threshold and the number of train iterations). Figure 2 shows the distributions of the damage types after splitting. Generally, though different countries have different damage type distributions, the combined distribution has a better balance among damage types. Also, using the origins of damage types as a stratified field seems sufficient since the train vs. evaluation distributions are somewhat similar.

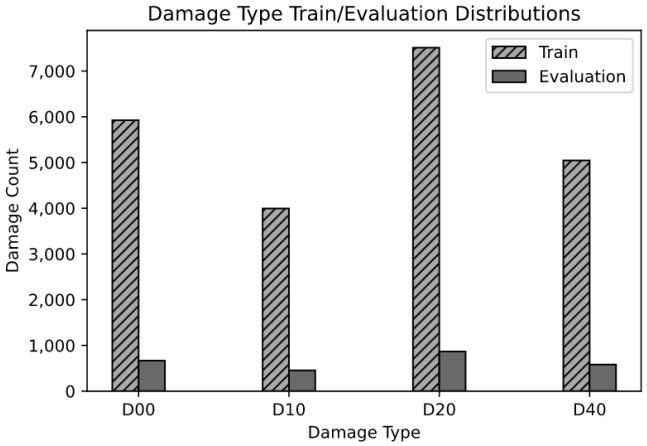

Fig. 2. Damage type distributions of the training dataset after splitting it into training and evaluation sets using country as stratify field. It is observable that the train and evaluation sets have relatively similar damage type distributions.

B. Base models

As discussed in Section III-B, most of the successful road damage detection and classification techniques trained using the Road Damage Detection dataset (version 2018) use the Faster R-CNN technique. Therefore, we first explore this approach to tackle this year's challenge. Instead of developing a Faster R-CNN model from scratch, we use Detectron2 [29] to speed up our development cycle. Detectron2 is Facebook AI Research's next-generation software system that implements state-of-the-art object detection algorithms.

It is also a common practice to use a base model pre-trained on a large image set (such as ImageNet [30]) as the feature extractor part of the network. Detectron2 provides many such base-models [31]. However, for Faster R-CNN, two commonly used base models are R101-FPN[1] and X101-FPN[2]. We select to explore these two pre-trained models because they have good Faster R-CNN box Average Precision (AP) compared to others. They are 42% and 43% on the pre-trained dataset (ImageNet), respectively. Though X101-FPN has better box AP on the ImageNet benchmark, it takes longer to train/predict and might be overfitting in some cases. That is the reason why we also explore R101-FPN.

Figure 3 depicts three main components of a Faster R-CNN model. They are the Backbone network, Region proposal network, and Box Head. The backbone network extracts features from the input image. In this case, we use the Feature Pyramid Network (FPN) types of the backbone. Therefore, it pulls the features at different scales for better predictions of anchor boxes of various sizes. The Region Proposal Network detects object regions from multi-scale features. It proposes the regions with objectness scores (how likely there is an object in a region) and the anchor deltas (centers and sizes relative to the picture size). The extracted features and box proposals are passed through an RoI Pooling (Region of Interest Pooling) layer to give standard inputs for the Box Head layer. Finally, Box Head is another neural network to predict the fine-tune bounding box locations and box classifications.

There are many hyper-parameters to be tuned while training a Faster R-CNN model. Thus, it is nearly impossible in terms of time and computation resources to explore all the configurations. Therefore, we first stick with the default and obvious, common-sense configurations. Specifically, we set *cfg.SOLVER.REFERENCE_WORLD_SIZE* (number of GPUs) to 2, both *cfg.SOLVER.IMS_PER_BATCH* (images per batch) and *cfg.DATALOADER.NUM_WORKERS* to 16, *cfg.SOLVER.BASE_LR* (base learning rate) as 0.00025, and *cfg.MODEL.ROI_HEADS.NUM_CLASSES* (number of classes) is 4 (as correspond to four different types of damages). All other configurations are kept as default from Detectron2. Interested readers can refer to Detectron2's documentation page[3] for further details about these default configurations.

[1]https://dl.fbaipublicfiles.com/detectron2/COCO-Detection/faster_rcnn_R_101_FPN_3x/137851257/model_final_f6e8b1.pkl

[2]https://dl.fbaipublicfiles.com/detectron2/COCO-Detection/faster_rcnn_X_101_32x8d_FPN_3x/139173657/model_final_68b088.pkl

[3]https://detectron2.readthedocs.io/modules/config.html

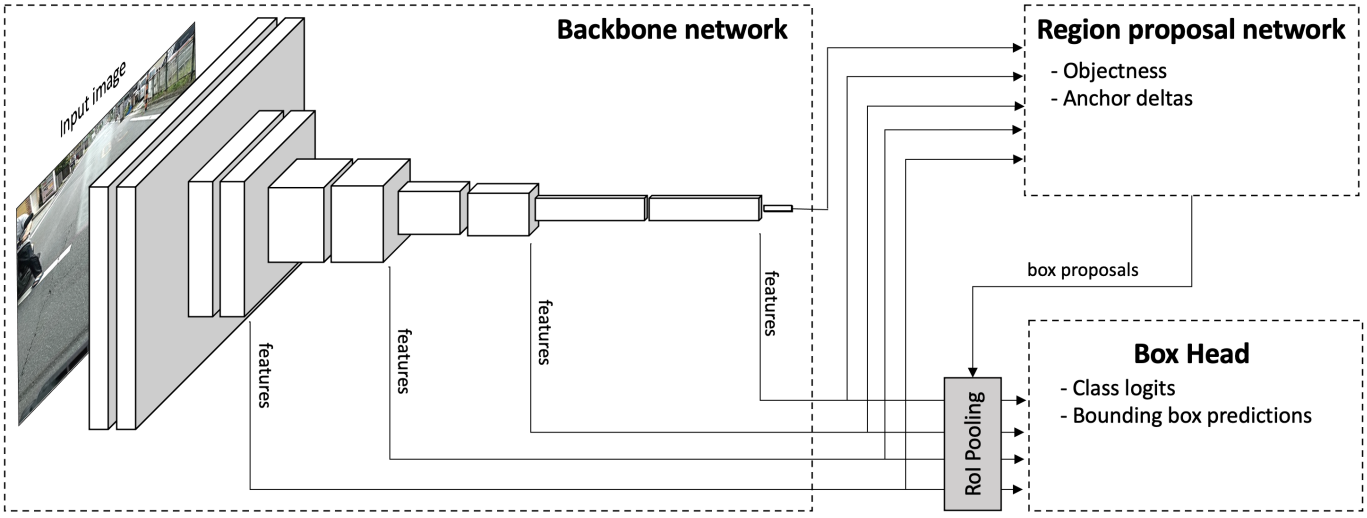

Fig. 3. The architecture of a Faster R-CNN model with three main components. The backbone network to extract features from the input image. The region proposal network to find box candidates with objectness scores and box deltas. The RoI Head to make the final box classifications and the bounding box predictions.

As shown in Table I, although it is faster to train a Faster R-CNN using R101-FPN as the base model, it has a slightly lower accuracy than using X101-FPN as the base model (52.85% vs. 54.25% on the evaluation set at the *test time prediction score thresholds* of 0.65 and 0.71, respectively). Specifically, it takes 0.82 seconds to train an iteration when using R101-FPN while it takes 1.67 seconds when using X101-FPN (using the specified configuration). The former also takes fewer iterations to converge than the latter (85,000 vs. 105,000). It takes time to do all the experiments with these models. Therefore, in the next sections, we only explore further experiments using the X101-FPN as the Faster R-CNN base.

C. Data augmentations

As discussed in Section IV-A, though the training dataset has a good number of images and damage types, they are imbalanced among damage types and are distributed differently for different countries. Therefore, besides the apparent augmentation techniques (such as image resizing and horizontal flipping, called 'default augmentations' hereafter), we also explore other augmentation techniques. Specifically, Maeda et al., [25] suggest that using GAN to generate synthetic damage types with fewer occurrences helps predict that specific type of damages (e.g., 'pothole' type of damages). Generating images using GAN often involves three processes: 1) generating a patch for damage, 2) finding places in an existing image to blend the patch, and 3) making the synthetic patch as natural to (as if it really belongs to) the container image as possible.

The synthetic damage patch generation step using GAN takes a long time to train. Additionally, we can manually place the patches to an existing image [25] or use another object detection model to detect the road areas and automatically position the artificial patches [21] to these areas. However, both of these approaches take time to complete. Therefore, in the following sections, we detail our strategies to quickly evaluate the efficiency of this augmentation method before developing and training complicated models for these tasks, in case this direction is promising.

Instead of using GAN to generate the synthetic damage patches, we randomly extract and select existing patches using the given ground-truth boxes. We also apply some simple augmentations to the selected patches (such as horizontal flipping, slight rotating, or scaling) to increase the varieties of the damage patches. Furthermore, we also randomly sample the locations from all existing damages of the same type to position the artificial patch into an existing image. The reason is that we assume that similar damage types would appear in similar locations. This heuristic does not always hold, but it would be fast and easy to implement.

Furthermore, instead of using Poisson blending [28] to place the synthetic damage patches into an existing image, we use a fast color transfer technique [32] to make the artificial patches look more natural to the placing image. Figure 4 shows an example of the resulted image. *D01* (green box) and *D10* (blue box) damage boxes are the real ones, while *D40* (yellow box) is the synthesized one. Though not all the artificial patches generated this way look natural or in the correct place, it helps to quickly generate augmented data and check if this augmentation strategy worth exploring.

Furthermore, in this case, we are not focusing on any specific type of damages but would like to balance the number of damages among the four damage types per country. Thus, we set different augmentation probabilities for different damage types in each country. Specifically, based on data exploration as shown in Figure 1, we set the augmentation probabilities as in Table II. Finally, we train the model (with the selected architecture described above with X101 as the base model) using the augmented data. It takes longer training iterations to

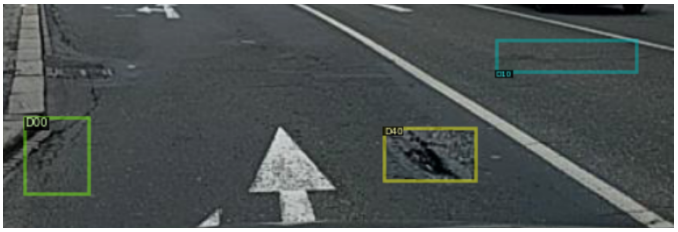

Fig. 4. Damage augmentation: *D00* (green-box) and *D10* (blue box) are the real damages. *D40* (yellow box) is the augmented one. This patch was randomly selected from existing *D40* damages then slightly modified and placed into a position randomly sampled from all positions of this crack type.

converge, and the resulted model does not perform better. This result might indicate that though individual countries may have different numbers of damages per type, the combined dataset (with three different countries) does not pose a big issue about damage type unbalancing (as shown in Figure 2).

TABLE II
AUGMENTATION PROBABILITIES FOR DIFFERENT DAMAGE TYPES AND DIFFERENT COUNTRIES.

Country	D01	D10	D20	D40
Czech	0.2	0.2	0.0	0.4
India	0.3	0.5	0.2	0.0
Japan	0.0	0.6	0.3	0.5

We also experiment with several other augmentation methods such as random cutout, random brightness/contrast, and cropping image augmentations, using *albumentations* package [33]. Specifically, for cutout augmentation, we randomly set up to 8 random areas with maximum sizes of 32×32 with the random probability of 0.5 (i.e., there is a 0.5 chance of augmenting for every one of the eight cutout areas). For brightness/and contrasts, we experiment both with a limit of 0.3 and a random probability of 0.5. Also, for cropping, we set the min and max sizes to 512 and 540, respectively. The cropping probability used is 0.1. However, all these augmentations take a higher number of training iterations to converge, and none of them help to increase the validation accuracy. Specifically, as shown in Table I, when we use these additional augmentations (i.e., default augmentations, artificial patches, random cutout, random brightness/contrast, and cropping), it takes 135,000 training iterations to converge instead of 105,000 for the X101 model with default augmentations. Moreover, its resulted accuracy is slightly lower (53.98% vs. 54.25%, respectively).

The cutout augmentation technique does not work might due to the reason that there are already built-in dropout layers in the Detectron2 Faster R-CNN implementation. The brightness/contrast technique does not help improve the performance, which might indicate that the training data itself already contains images with different light and weather conditions. In other words, they already have images with different brightness/contrast levels. Lastly, the cropping augmentation technique does not work might due to the reason that Faster R-CNN uses only the regions of interests of each training image (i.e., ground-truth boxes) rather than the entire image [20].

We explore also test time augmentation (TTA). Specifically, we apply flipping, resizing, and brightness/contrast augmentation techniques. We only apply horizontal flipping for flipping augmentation since the vertical flipping does not make sense in this case. For resizing, we use the following sizes (400, 500, 600, 700, 800, 900, 1000, 1100, 1200). TTA takes a longer time to make predictions because they have to make predictions for several augmented images, then combine them to produce final predictions for each image. However, neither of these approaches helps to improve prediction accuracy. Specifically, TTA reduces the F1 score for the model with X101 as the base and default augmentations to 50.73% (from 54.25% when not using TTA).

D. Other hyperparameters

Batch normalization is one of the breakthroughs in deep learning. It allows faster and more stable training by making the output distribution from one layer stable before forwarding to the next layer. This strategy also helps gradient descent by avoiding vanishing gradients. It normalizes the previous layer's output by subtracting the empirical means over the batch divided by the observed standard deviations. Therefore, there are options to change the pixel means and standard deviations over the three image channels (i.e., Red, Green, and Blue).

Since Detectron2 recommends not to change the standard deviations, we look into the pixel means (cfg.MODEL.PIXEL_MEAN) from all the images in training set as $[122.190, 122.639, 117.788]$ (in Blue, Green, and Red channels, respectively) and use them instead of the default values (generated from ImageNet dataset) as $[103.530, 116.280, 123.675]$. However, this approach does not help to enhance performance. This result might indicate that the calculated means for this dataset and those from ImageNet are not very different. Another indication would be the base model was trained with the default means and standard deviations, so changing them impacts the extracted features from the base model used.

Furthermore, Faster R-CNN has a module to generate anchor boxes. These anchor boxes are generated with different sizes and ratios. Therefore, we explore our data to find appropriate box sizes and ratios. Specifically, Figure 5 shows the histogram of the areas of all ground-truth bounding boxes for the training set's road damages. It is observable that the areas are distributed mostly around 0 to 400 squared pixels. Therefore, we set the anchor generator sizes (*cfg.MODEL.ANCHOR_GENERATOR.SIZES*) to $[[32, 64, 128]]$ instead of the default $[[32, 64, 128, 256]]$.

The ratios of the anchor boxes are calculated by their heights over their widths. Therefore, we also explore the $height/width$ distribution of all the ground-truth bounding boxes. It is observable from Figure 6 that a high number of the ratios are distributed at the lower end. Therefore, we set the ratios

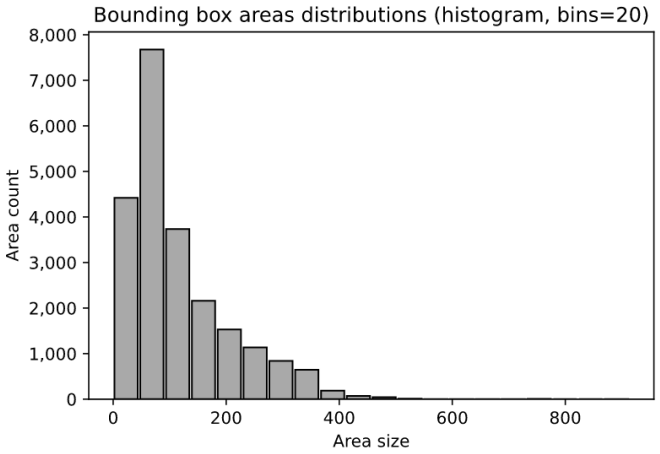

Fig. 5. Bounding box area distributions using a histogram with the number of bins as 20 (range of a bin is approximately 45.6). It is observable that the areas distributed mostly around 0 to 400 squared pixels.

(cfg.MODEL.ANCHOR_GENERATOR.ASPECT_RATIOS) to these values $[[0.1, 0.5, 1.0, 1.5]]$ instead of the default $[[0.5, 1.0, 2.0]]$ ratios. Training our model with these parameters does not help to improve the results much. However, it helps make the learning process converges faster. Specifically, as shown in Table I, it converges around the iterations 70,000 instead of 105,000 and the accuracy is at 54.51%, which is similar to that of the model with X101 base and default configurations. These results indicate that specifying appropriate sizes for the anchor boxes helps the learning speed. However, the model is complicated enough to learn the boxes though the candidates are not very close to the predicted ones.

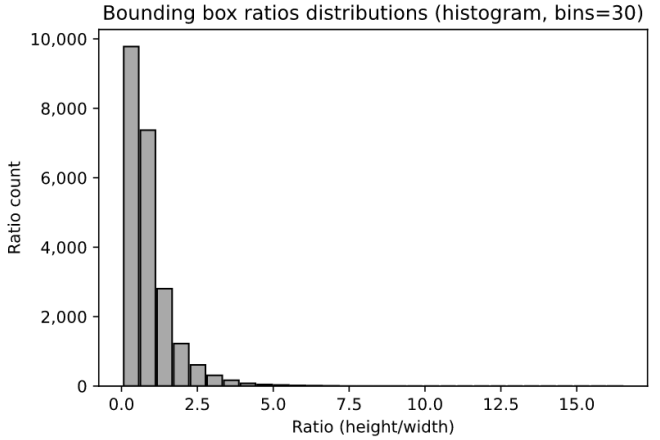

Fig. 6. Bounding box ratios (height/width) distributions with the number of bins as 30 (the width of a bin is approximately 0.55). It is observable that this distribution is skewed toward the lower end.

E. EfficientDet model

As discussed, the object detection research field is emerging, and related techniques evolve consistently. Specifically, Google Brain team recently published EfficientDet [34], which claims to achieve a state-of-the-art result on Common objects in context (COCO) test-dev [35]. Therefore, we also implement and train a model using EfficientDet. However, the prediction result is not as good as Faster R-CNN. Though it takes a similar amount of time to train, it does not produce a better prediction performance. This result might indicate either EfficientDet is not suitable for this dataset or need more experiments to tune its parameters for this specific case.

V. Evaluations and Suggestions

We also visualize the predicted bounding boxes with corresponding labels and scores to qualitatively evaluate the results. In general, predictions and ground-truth match reasonably well. However, we also discover several discrepancies and found some wrong/missing ground-truth bounding boxes. Figure 7 shows a few of these discrepancies. The red boxes are ground-truth, and the blue boxes are predicted ones. The predicted boxes also have corresponding labels and prediction probabilities (ranging from 0 to 1.0 exclusively as low to high confidence of having a road crack within the box). We only show three examples in this case, and we also crop and keep only the lower parts of these pictures due to space limitations. The pixel numbers (on the axes) allow the interested readers to identify these regions in the original images correspondingly. For clarity of these pictures, we recommend interested readers to check these pictures in their original sizes from the training folder using the image file names listed on top of the pictures.

Specifically, the first picture is *Japan_008976*, while the *D00* prediction matches the ground-truth, another detected damage of type *D20* is obvious but is not in the annotations provided by the dataset. On the other hand, there are two horizontal cracks detected with high confidences in the picture *Czech_001957*, but there are no annotations for these in the training set. Finally, there is a pothole (*D40*) damage detected in *Japan_012722* without a corresponding ground-truth. It is also worth noting from this picture (*Japan_012722*) that there are currently two overlapping bounding boxes with the same damage type (*D20*), but one with 0.99 and another with 0.60 confidence scores, respectively. The predictive score threshold and the non-maximum suppression step will remove the one with a lower prediction score. Thus, the predicted one bounding box for *D20* matches the ground-truth one.

In many cases, especially in darker light conditions, machine learning models can perform better than humans. For instance, Figure 8 shows the picture *India_009632* from *test1* set. Our model predicts that there is a *D02* type of damage in the picture (the red box in panel (A)). However, this is not clear to humans. We might think that was a wrong prediction as one can visibly check in the top panel. However, it turns out to be a correct prediction when we set the brightness of this picture to 150 (e.g., using Photoshop) and zoom into this region (200%), as shown in panel (B). Similar story happens to image *Japan_012412* and the detected *D00* damage type in the panels (C) and (D).

These examples imply that the human manual labeling approach might not be sufficient for this type of dataset. In

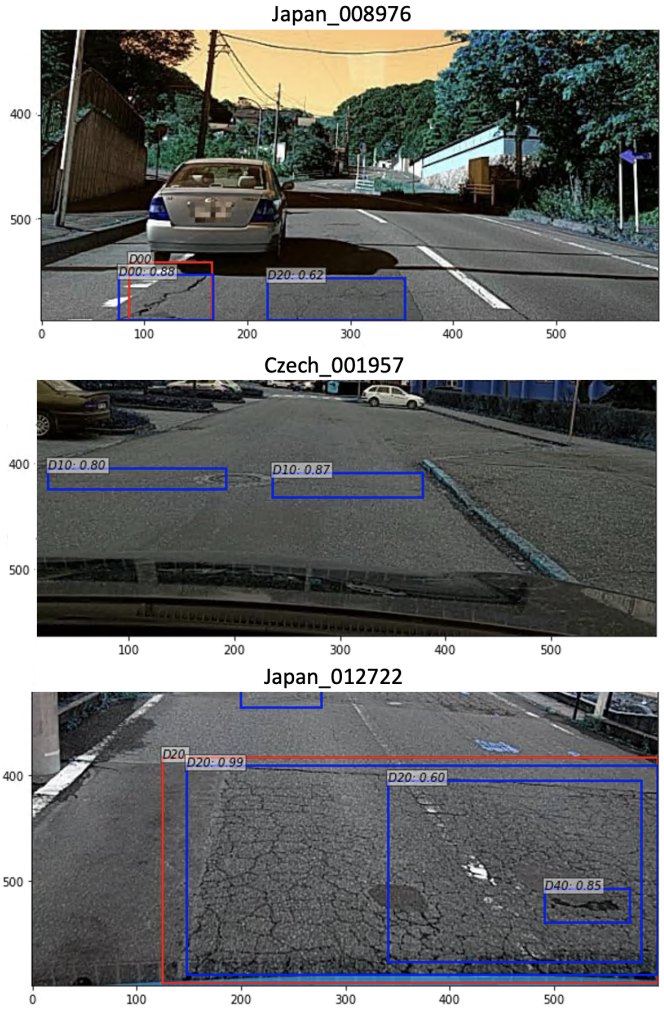

Fig. 7. Examples of missing bounding boxes and labels in the training set. In this case, the red boxes are ground-truth, and the blue boxes are the predicted ones. Both have corresponding labels (for damage types), and there is also the predicted scores for the predictions. It is observable that there are road cracks predicted, but they are missing in the evaluation set.

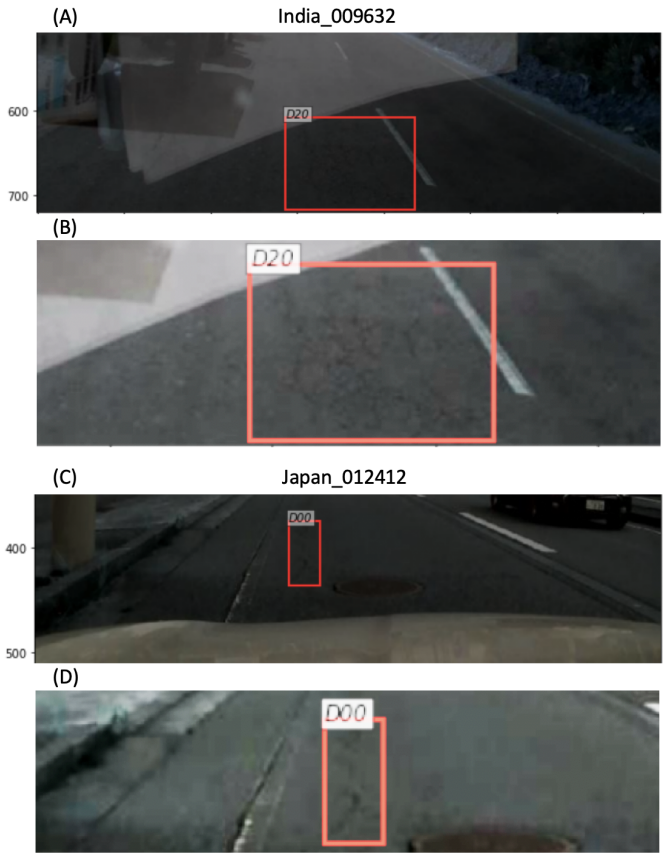

Fig. 8. Examples of the case that the machine learning model outperforms humans. Panel (A) shows that our machine learning model detects a *D02* type of damage in the image *India_009632*, and it is not visible to humans. It is only perceptible when we change the brightness of the picture and zoom into the region, as in panel (B). Similar story happens to the image *Japan_012412* for damage type *D00* in panels (C) and (D).

other words, there should be a combination of both human and machine learning supports for this task. This missing label issue (due to human labeling) for the training set would be acceptable. However, it would be inaccurate to evaluate the performance of models if it happens to the testing sets. Therefore, we suggest that the test sets should be brightened and zoomed before the human labeling process. After labeling, these images can then be converted back to their original brightness and sizes.

Deep learning needs a massive amount of labeled data, and manual labeling of road damages takes so much time and is error-prone. The reasons for errors might due to different light conditions, precisions of the bounding boxes are difficult to set, or even the confusion between damage types. Kluger et al., [21] suggest using a learned model to predict the bounding boxes and classify damage types of the unlabeled, collected images. These predictions then undergo the human manual checks to validate/change the bounding boxes or the labels. However, it still takes time, and there is a lot of room for errors with human manual label checks. Figure 9, shows an example of such mistakes. Image *Japan_002970* (from the *train* set) has a wrong *pothole* damage bounding box. This wrong label is obvious to humans and less clear to machine learning models. This type of error also indicates that several labels in this current training set were generated automatically and did not get validated carefully by humans. This error would also indicate that the labeling process depends on one model learned on the initial limited amount of data.

On the other hand, in a recent study, Xie et al., [36] claim to have state-of-the-art ImageNet classification results using self-learning with noisy student technique. We suggest using this approach with the following steps. First, train a model on existing images with labeled bounding boxes and classes. Use it as a teacher to generate annotations for other unlabeled images. Combine the newly predicted annotations (with higher confidences) with the existing annotations, then train a larger model called a student model. Use this learned student as a teacher and repeat the process. This approach incrementally

Fig. 9. An image (*Japan_002970*) with wrong *pothole* damage bounding box in the training set for Japan. This mistake is obvious to humans and less clear to machine learning models. Thus, it might indicate that a machine learning model generates this label, and the error was slipped through the human quality check step.

increases the labeled data instead of training on a single model then uses that single model to predict labels as in the previous approach.

It would always be useful to finally pass the pseudo, predicted labels through another human inspection to validate or adjust the bounding boxes and labels as a quality check. However, it's worth noting that one should also change the brightness and zoom into the predicted regions for better quality checks if they are not apparent to humans. These cases can be done manually or detected (using overall light conditions) automatically adjust the pictures to help the validation process.

VI. Selected Model, Results, and Limitations

We selected Faster R-CNN with X101-FPN and Detectron2's default configurations as the architecture to tackle the tasks of this competition. Though this approach (named as 'X101' in Table I) takes a little longer to train and slightly lower accuracy compared to the model 'X101+SRs' (the same model but uses custom ratios and sizes for the anchor boxes), it is general and is easier to implement. Thus, it is more practical and easy to transfer to different datasets from different countries in the future (as one of the aims of this competition). This selected model (with the parameters described above) results in F1 scores of 51.0% and 51.4% for the *test1* and *test2* sets of the challenge, respectively. The F1 scores on the test sets that are slightly lower than that on the evaluation set (as shown in Table I) might indicate that the two test sets and the train set (thus, its split for the evaluation set) may have different data distributions. As a matter of fact, these differences in data distributions are also confirmed by the organizer [37].

These scores ranked 11^{th} among 12 shorted-list winners (out of 121 participants). Interested readers can refer to the summary of these winning approaches in this summary paper [37] from the organizer. It is observable that these state-of-the-art solutions, summarized in this paper, have base scores similar to ours. However, the higher final scores of these higher ranked solutions are mostly based on ensembles. For instance, one solution trains 30 models with different hyper-parameters and ensembles them to get the top results. As discussed, we do not attempt ensemble direction since these are not practical in the real world because they do not scale well.

All in all, the low F1 score is arguably acceptable due to several practical constraints and the issues with the ground-truth labels in the training sets, as detailed in the previous sections. However, the selected approach still has the main limitations as using Faster R-CNN with X101-FPN is slower to train and has a longer prediction time than other model types such as YOLO and SSD.

Furthermore, deep learning models are black-box in nature [16]. Thus, in the future, we would also like to research and explore the features extracted at the intermediate layers of the learned models. Visualizations of these extracted features help explain how and why this model performs on accurate and wrong predictions to improve its performance [38].

VII. Conclusion

This work explores different state-of-the-art object detection methods and their applicability for road damage detection and classification tasks. Specifically, we experiment with Detectron2's Faster R-CNN implementation with different base models and configurations using the Global Road Damage Detection Challenge 2020 dataset. We also examine other state-of-the-art object detection methods and various techniques like training time augmentations, testing time augmentations, context information, and post-processing. However, these methods are not suitable for the damaged road objects, and their effects are not satisfactory. In other words, the results indicate that Faster R-CNN with X101-FPN base model and Detectron2's default configurations produce good prediction results for these tasks (F1 score of approximately 51.0% for both test sets) and is also general enough to be used in different countries. We also visualize and qualitatively evaluate the existing labeling quality and suggest using the noisy student approach to improve the road damage labeling process. In the future, we suggest visualizing the intermediate features extracted by this model. These visualizations will provide insights bout how and why the model makes its predictions—these profound understandings of how the model works will enable modification of the model for better performance.

The source codes of the experiments are available at the Github page of this project: https://github.com/iDataVisualizationLab/roaddamagedetector.

REFERENCES

[1] V. Pham, N. V. Nguyen, and T. Dang, "Scagcnn: Estimating visual characterizations of 2d scatterplots via convolution neural network," in *Proceedings of the 11th International Conference on Advances in Information Technology*, ser. IAIT2020. New York, NY, USA: Association for Computing Machinery, 2020. [Online]. Available: https://doi.org/10.1145/3406601.3406644

[2] H. Maeda, Y. Sekimoto, T. Seto, T. Kashiyama, and H. Omata, "Road damage detection using deep neural networks with images captured through a smartphone," *arXiv preprint arXiv:1801.09454*, 2018.

[3] D. Arya, H. Maeda, S. K. Ghosh, D. Toshniwal, A. Mraz, T. Kashiyama, and Y. Sekimoto, "Transfer learning-based road damage detection for multiple countries," 2020.

[4] IEEE BigData 2020, "Global road damage detection challenge 2020," https://rdd2020.sekilab.global/, accessed: 2020–10-16.

[5] K. Tong, Y. Wu, and F. Zhou, "Recent advances in small object detection based on deep learning: A review," *Image and Vision Computing*, p. 103910, 2020.

[6] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[7] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[9] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[10] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[11] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[12] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7263–7271.

[13] ——, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[14] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020.

[15] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*. Springer, 2016, pp. 21–37.

[16] D. D. Le, V. Pham, H. N. Nguyen, and T. Dang, "Visualization and explainable machine learning for efficient manufacturing and system operations," 2019.

[17] V. Pham, N. V. Nguyen, and T. Dang, "Scagcnn: Estimating visual characterizations of 2d scatterplots via convolution neural network," in *Proceedings of the 11th International Conference on Advances in Information Technology*, 2020, pp. 1–9.

[18] Y. J. Wang, M. Ding, S. Kan, S. Zhang, and C. Lu, "Deep proposal and detection networks for road damage detection and classification," in *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, 2018, pp. 5224–5227.

[19] W. Wang, B. Wu, S. Yang, and Z. Wang, "Road damage detection and classification with faster r-cnn," in *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, 2018, pp. 5220–5223.

[20] A. Alfarrarjeh, D. Trivedi, S. H. Kim, and C. Shahabi, "A deep learning approach for road damage detection from smartphone images," in *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, 2018, pp. 5201–5204.

[21] F. Kluger, C. Reinders, K. Raetz, P. Schelske, B. Wandt, H. Ackermann, and B. Rosenhahn, "Region-based cycle-consistent data augmentation for object detection," in *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, 2018, pp. 5205–5211.

[22] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.

[23] C. Reinders, H. Ackermann, M. Y. Yang, and B. Rosenhahn, "Object recognition from very few training examples for enhancing bicycle maps," in *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 1–8.

[24] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2223–2232.

[25] H. Maeda, T. Kashiyama, Y. Sekimoto, T. Seto, and H. Omata, "Generative adversarial network for road damage detection," *Computer-Aided Civil and Infrastructure Engineering*, 2020.

[26] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[27] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," *arXiv preprint arXiv:1710.10196*, 2017.

[28] P. Pérez, M. Gangnet, and A. Blake, "Poisson image editing," in *ACM SIGGRAPH 2003 Papers*, 2003, pp. 313–318.

[29] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, "Detectron2," https://github.com/facebookresearch/detectron2, 2019.

[30] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[31] Detectron2, "Detectron2 model zoo," https://github.com/facebookresearch/detectron2/blob/master/MODEL_ZOO.md, accessed: 2020-10-16.

[32] Rosebrock, Adrian, "Super fast color transfer between images," https://www.pyimagesearch.com/2014/06/30/super-fast-color-transfer-images/, accessed: 2020-10-16.

[33] A. Buslaev, V. I. Iglovikov, E. Khvedchenya, A. Parinov, M. Druzhinin, and A. A. Kalinin, "Albumentations: Fast and flexible image augmentations," *Information*, vol. 11, no. 2, 2020. [Online]. Available: https://www.mdpi.com/2078-2489/11/2/125

[34] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: Scalable and efficient object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 10 781–10 790.

[35] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*. Springer, 2014, pp. 740–755.

[36] Q. Xie, M.-T. Luong, E. Hovy, and Q. V. Le, "Self-training with noisy student improves imagenet classification," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 10 687–10 698.

[37] D. Arya, H. Maeda, S. K. Ghosh, D. Toshniwal, H. Omata, T. Kashiyama, and Y. Sekimoto, "Global road damage detection: State-of-the-art solutions," 2020.

[38] T. Dang, H. Van, H. Nguyen, V. Pham, and R. Hewett, "Deepvix: Explaining long short-term memory network with high dimensional time series data," in *Proceedings of the 11th International Conference on Advances in Information Technology*, 2020, pp. 1–10.