Detection of Japanese Knotweed Beside the Road using Deep Learning

Siddharth Surya
Embedded Systems
s.surya@student.utwente.nl

***Abstract*—Invasive Alien Plant Species (Japanese Knotweed) are a threat to biodiversity. Its monitoring can help conserve the native flora and fauna. Japanese Knotweed habitat is generally along the roadways and, its identification, can prevent its spread. The research discusses the detection of the invasive Japanese Knotweed plant using a mobile phone camera mounted on a bicycle. It mainly focuses on object detection models under two categories: (1) Generic and (2) Few-shot. Generic object detection has been used to detect invasive plants. However, a lack of pre-labeled datasets makes this approach expensive and time-consuming. Moreover, the object detection model considering the plant phenological cycle further increases the requirement for annotated images. Thus, the few-shot object detection algorithm offers an alternative approach with limited annotated images. However, there is no previous study on its performance for invasive plant species (Japanese Knotweed). Thus, this study aims to bridge these gaps. It considers the creation of a realistic dataset for detecting invasive Japanese Knotweed plants using a mobile phone camera mounted on a bicycle. A performance comparison between the extensively used general object detection i.e, Faster RCNN, state-of-art YOLOv7, and few-shot state-of-art 'DeFRCN' is performed under PASCAL VOC settings of 1, 2, 3, 5, 10 images and complete dataset. Resolution & support set study concerning DeFRCN is also discussed. It was found that under limited training images (1, 2, 3, 5, 10), the generic object detection overfits and few-shot object detection model with data augmentation offers 5.33, 4.83, 6.45, 7.31, 8.30 $\mathrm{mAP}_{50}{}^{\mathrm{test}}$ respectively. YOLOv7 offers the highest $\mathrm{mAP}_{50}{}^{\mathrm{test}}$ with a complete dataset, which is 32.6. On the contrary, the Faster RCNN has a large false positive and is not robust. Overall, the research focuses on contributing to the conservation of native flora and fauna.**

***Index Terms*—Invasive Species, Japanese Knotweed, Deep CNN, Roadside, Few-Shot Object Detection, Mobile Camera**

I. Introduction

The invasive Alien Plant (IPAS) poses a severe threat to the native environment and causes damage worth billions of euros to the European economy every year [1]. IAPS are non-native plants that are accidentally or intentionally introduced in a new geographical area. One such plant is the Japanese Knotweed (Polygonum Cuspidatum) which is listed 37th on the '100 of the World's Worst Invasive Alien Species' by IUCN. It is a herbaceous plant that is native to Japan that was brought to Europe as a decorative plant, and to stabilize soil in the coastal areas. However, when water is infested with its fragments, it relocates to a new area i.e, riparian zone and communication routes that were previously uninhabited by it [2]. Here, it grows into dense stands, displacing native flora and fauna [3]. Moreover, its powerful steam damages buildings, pipes, and roads, incurring substantial economic loss. Thus, detecting Japanese Knotweed can aid in containing its spread.

Researchers have used various techniques (Object-Based Image Analysis [4] [5] & generic object detection algorithms) to detect invasive species. And, in general, the detection approach can be divided into a pipeline consisting of three major steps: (1) Image Acquisition (2) Image Pre-processing (3) Image Classification & Object Detection [6].

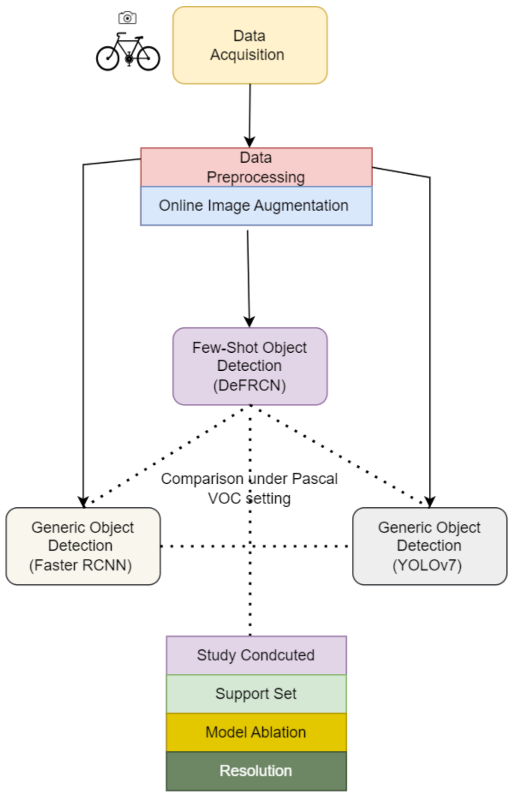

Data Acquisition
Data Preprocessing
Online Image Augmentation
Few-Shot Object Detection (DeFRCN)
Comparison under Pascal VOC setting
Generic Object Detection (Faster RCNN)
Generic Object Detection (YOLOv7)
Study Condcuted
Support Set
Model Ablation
Resolution

Fig. 1: Simplified overview of the research

For image acquisition of invasive species, satellites, UAVs, and cameras mounted on vehicles have been used [4] [7] [8]. While each acquisition method has advantages, it also has limitations. For instance, satellite imagery is often accompanied by multi-spectral data to distinguish between vegetation and its types. But, high-resolution data are costly and are restricted to a specific area. The medium and low-resolution spectral data is freely accessible, but some fine-scale image characteristics

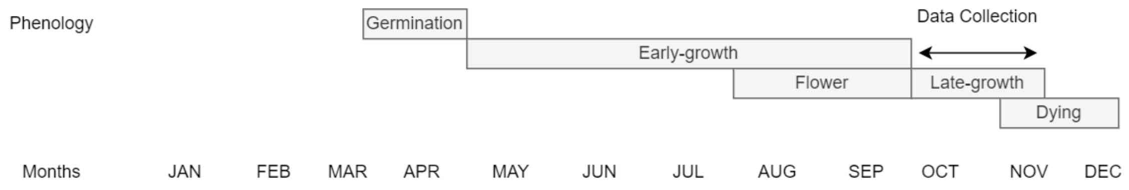


Fig. 2: Phenological cycle of Japanese Knotweed. In a few locations, flower and early-growth phase extends till October.

are lost, and invasive species that occur in small patches are difficult to identify [9]. Therefore, it offers promising results for detecting long and dense stands (> 10 m^2), but it misses the localized stands of Japanese Knotweed (< 2 m^2) [10] [5]. Moreover, satellite imagery frequently suffers from a lack of coverage due to clouds and sensor problems. Thus, irregular temporal image data is another challenge. Similarly, drones can cover a relatively larger geographical area but have a limited flight time as it requires frequent charging. Secondly, it violates the privacy right of the citizen by capturing images without drawing their attention [11]. Thirdly, weather condition restricts the deployment of drones. [12].

This research focuses on a mobile phone camera mounted on the bicycle for acquiring images of Japanese Knotweed beside the road. Not only it is cost-effective compared to the other acquisition methods, but geotagged crowdsourced mobile images and videos can also be utilized. Additionally, it will increase the reachability over long distances and isolated areas, discovering new patches of Japanese Knotweed. Thus, it offers an unmatched spatial resolution at the country level and high temporal resolution.

But, Japanese Knotweed detection besides the road is challenging because of its similarity in color with the other plants. Also, it appears dissimilar (color and texture) under different illumination and during its growth phase. Moreover, the plant's proximity to the road, varying camera resolutions, motion blur & noise in the image increases the difficulty of its detection [13]. Nevertheless, researchers have extensively used deep learning for identifying invasive species because of its strong feature learning capabilities. For addressing other challenges, image augmentation is generally employed as part of image pre-processing in the detection pipeline. This ensures that the model generalizes to the realistic environment [14] [15] [16] [17].

Although typical deep learning methods have achieved good performance in detecting invasive species, it requires hundreds and thousands of annotated images (11,240 images of Solanum Rostratum Dunal plants [18], 3826 images of Hydrangea [16], 2500 images of different invasive plant species each [19]). Annotating images is often expensive and time-consuming. This is a more prominent problem in the case of object detection, where dense labeling of bounding boxes in each image is needed. Moreover, there is a lack of pre-labeled datasets for fine-grained object detection applications such as the detection of Japanese Knotweed. Additionally, plants change their appearance throughout their phenological cycle as shown in figure 2, and failure to account for these changes in an object detection model would result in poor generalization to plants at different stages of growth. This increases the cost of data collection and the development of a robust model.

Thus, a few-shot object detection model can offer a practical solution for detecting Japanese Knotweed under a limited dataset. It reduces costs in dataset creation and model development.

This research aims to bridge a gap in detecting invasive species, particularly Japanese Knotweed under a limited dataset using a state-of-art few-shot object detection algorithm. Researchers have used generic object detection for identifying invasive species, but to the best of my knowledge, there exists no literature on few-shot object detection of invasive plant species. Additionally, there has been no work on detecting Japanese Knotweed from the videos/photos taken from the roadside on a bike. Also, a comparison has been performed between both methods under standard few-shot PASCAL VOC setting of 1, 2, 3, 5, 10 shot and complete dataset. Thus, these novelties will contribute in conserving native flora and fauna.

The thesis overview is briefly summarized in the figure 1. And, the following question has been formulated to conduct the research:

How can object detection be performed for Japanese Knotweed with a limited annotated dataset under varying conditions such as distance, lighting, illumination, angles, resolution, and plant phenological cycle?

The above research question is further divided into the following sub-research questions:

1) What strategies can be used to create and annotate a diverse and representative dataset for object detection of Japanese Knotweed?
2) What are the techniques for object detection of Japanese Knotweed? It is further divided into three sub-research questions:
    a) How does a general object detection model with a pre-trained weight and fine-tuning on a limited annotated dataset perform on the detection of Japanese Knotweed? Can data augmentation techniques improve its performance?

b) How does the performance of a few-shot object detection model, trained with pre-trained weights and fine-tuned on a limited annotated dataset, compare to (a) in detecting Japanese Knotweed?

c) How does the state-of-art generic object detection model with a pre-trained weight and fine-tuned on a limited annotated dataset compare to (a) and (b) in the detection of Japanese Knotweed?

To answer the above questions, mean average precision mAP_{50} on the test set, validation loss curve (where applicable), and visual inspection of the model's prediction on the test image will be used as evaluation metrics for all object detection models. The validation loss curve is essential in understanding the model's performance because the dataset is limited and models are susceptible to overfitting. Visual inspection provides an intuitive interpretation of false positives. Moreover, predictions on other classes are also visualized to examine the model's robustness.

The structure of the research paper is as follows: Section 47 provides a concise overview of the selected object detection algorithms, along with the rationale behind their selection. Section III, material and methods highlight dataset creation, cleaning, organization, and data augmentation. Section IV experimental setup, mentions the network training procedure and provides results from all the models. Section V compares and discusses the result. Finally, the conclusion & future scope has been mentioned in section VI.

II. Background

This section provides a concise overview of two generic and one few-shot object detection algorithm. Furthermore, the scientific reasoning behind selecting each algorithm has also been mentioned under respective sections.

A. Generic Object Detection

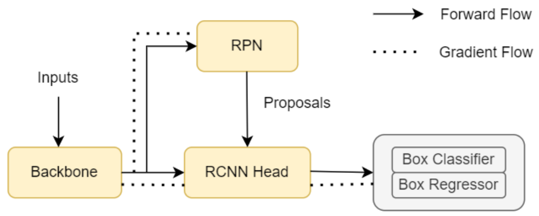


Fig. 3: Simplified Faster RCNN architecture

1) **Faster RCNN:** Faster RCNN is one of the most popular and widely used object detection algorithms. Previous studies (Milkweed detection beside road [20], Eupatorium Adenophorum detection [21]) have also used Faster RCNN for detecting invasive species. Secondly, most of the few-shot learning algorithms based on transfer learning and fine-tuning (including the selected DeFRCN algorithm used in this research) make changes to its structure and adapt it to work on a limited dataset [22] [23] [24]. Thus, comparison with Faster RCNN would give a better analysis of its drawbacks on limited dataset.

In Faster RCNN, the input image is first fed to the backbone (CNN) to obtained single scale features following which two stages are applied:

- The obtained features are fed to region proposal network (RPN) for extracting object proposals, i.e, bounding box that may contain an object. The object proposals are predicted at a predefined aspect ratio (Anchor Box), which is refined by regression head. Further, Non-Max Suppression (NMS) removes low quality and redundant proposals.
- Fast RCNN extracts a fixed-size feature map using a ROI Pool for the remaining proposed bounding box. It is then fed to a ROI head for predicting object class and refining bounding box coordinate. Additionally, NMS is used again to remove redundant and low-confidence predictions [25].

2) **YOLOv7:**

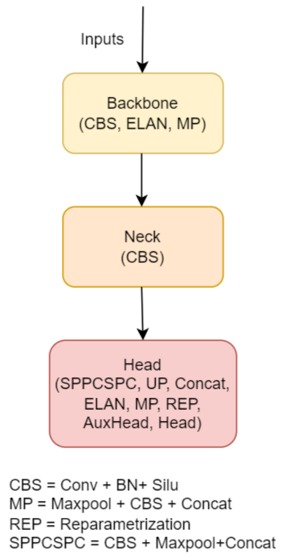

Inputs
Backbone
(CBS, ELAN, MP)
Neck
(CBS)
Head
(SPPCSPC, UP, Concat, ELAN, MP, REP, AuxHead, Head)
CBS = Conv + BN+ Silu
MP = Maxpool + CBS + Concat
REP = Reparametrization
SPPCSPC = CBS + Maxpool+Concat

Fig. 4: Abstract YOLOV7 architecture

YOLOv7 is the current state-of-art in generic object detection [26]. Therefore, it can establish a benchmark for comparison with Faster RCNN. Moreover, to the best of my knowledge, no study on its performance has been performed on a limited dataset. The performance comparison with few-shot learning model may provide insight to the researcher.

YOLOv7 is a single-stage object detector which gives a high inference rate compared to two-stage detectors (Faster RCNN). This is because it performs object classification and bounding-box regression in a single forward pass through the network, eliminating explicit region proposal generation and its subsequent refinement. YOLOv7 has different variations, which prioritize either accuracy or inference. The maximum inference speed it offers is 160 frames per second, and the maximum mAP$_{50}$ is 74.7.

YOLOv7 has made several architectural changes compare to the previous versions to improve detection speed and accuracy. The first change is using Extended Efficient Layer Aggregation Network (E-ELAN) as backbone instead of the Darknet as its predecessors. E-ELAN uses expand (allows adding more filter to layers to learn complex features), shuffle (shuffles feature map), and merge cardinality (combines the output of multiple layers at different resolutions) and enhances the learning ability of the network by preserving the original gradient path. Second, Researchers generally use a Network/Neural Architecture Search (NAS) tool to address the problem of model scaling for a particular device deployment. It provides optimal scaling factor for resolution, width, depth, and stage (the number of feature pyramids). YOLOv7 enhances the scaling by using a compound scaling mechanism, which is a coherent scaling of width and depth parameters. Third, YOLOv7 also uses re-parameterization planning (RP) that averages various models to create a final model offering a robust performance. Lastly, the architecture's head component, utilizes a multi-head concept. The primary head is responsible for the final classification, while the auxiliary heads contribute to the training process in the middle layers.

B. Few-Shot Object Detection

Few-shot object detection aims at designing a model that can successfully operate in a limited data regime. Thus, following the Wang approach [27], three state-of-art few-shot approaches were studied. These are classified under three categories: (1) Data: Uses external semantic information (2) Model: Learns support and query relation (3) Algorithm: Refines parameter update strategy

1) **Data Approach (FADI):** FADI (Few-shot object detection via association and discrimination) [28] mentions that the transfer learning approach for few-shot object detection gives poor classification accuracy. This is due to the scattered feature space of the novel class, which violates inter-class separability between base and novel, leading to confusion in classification. Thus, it proposes 2 steps fine-tuning framework to create a discriminative feature space for the novel class: (1) Association (2) Discrimination

   In the association step, it associates the novel class with the base class based on Lin similarity (measures semantic similarity between two words). The novel class align its feature space with the respective base class and become naturally separable from other base class. Thus, it becomes an intra-class classification problem that is handled by the discrimination framework. In the discrimination, it disentangles the base and novel classes to remove confusion between them. It does so by extending the Faster RCNN architecture with two separate fully convolutional layers (base pre-trained & novel associated) and a classifier dedicated to base and novel class. FADI offers the highest mAP$_{50}$ in data approach with 63.2 considering 10 shot as a reference.

2) **Model Approach (Meta-DETR):** Meta-DETR mentions that most existing state-of-art literature is built on Faster-RCNN for solving the few-shot detection problem. But, the accuracy suffers because of two drawbacks: (1) Low-quality region proposal for novel class because of less availability of novel data (2) Most methods take one support class at a time and misses the relation among different support class. This limits the ability to distinguish a similar class and to generalize from the related class. Meta-DETR on contrary, works on the image level (using deformable DETR object detection). Moreover, Meta-DETR attends multiple support classes at one go and captures inter-class correlation for reducing the misclassification of a similar class. It offers the highest mAP$_{50}$ in model approach with 63.6 considering 10 shot as a reference.

3) **Algorithm Approach (DeFRCN):** Decoupled Faster RCNN provides state-of-art performance under low data constrain by improving conventional Faster RCNN architecture. The first problem it identifies with Faster RCNN is that RPN and RCNN shared the same backbone, with loss being jointly updated as mentioned below:

$$L_{total} = (L_{rpn}^{cls} + L_{rpn}^{reg}) + \eta \cdot (L_{rcnn}^{cls} + L_{\text{rcnn}}^{reg}) \quad (1)$$

   But they have different goals. RPN is class independent and RCNN is class relevant. This mismatch of goals reduces the classification ability. Thus, it offers a decoupling mechanism between three modules of Faster-RCNN: backbone, RPN, and RCNN such that one doesnt dominate over another using Gradient Decoupled Layer (GDL) as shown in figure 5 (Top image). GDL transforms the feature maps from the backbone into two distinct feature map (using affine transformation) during forward propagation for RPN and RCNN. During backward propagation, the gradient is multiplied with the decoupling constant. This allows the model to optimize the weights for each module independently and achieve decoupling between the two. With the decoupling constant from RPN being zero, the backbone is only updated based on the gradients from the RCNN, which are more relevant to the target domain (Japanese Knotweed). Thus, it tries to improve low quality region proposal as mentioned by Meta-DETR.

   The second problem it identifies is that the RCNN box classifier requires a translation invariant (features that

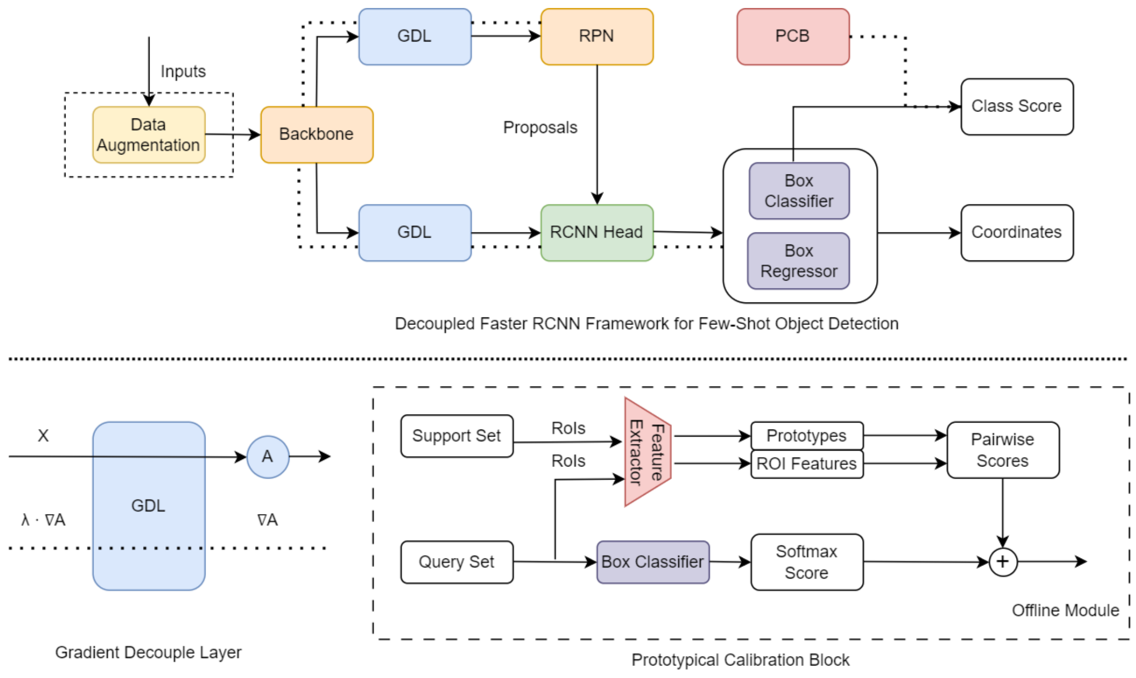


Fig. 5: Architecture of DeFRCN for few-shot object detection with GDL and PCB. 'A' denotes the affine transformation to the feature map 'X' in the GDL. PCB is trained offline and utilized during the inference. Data Augmentation is added to the existing architecture to increase its performance.

remain the same under translation) feature, while its box regressor requires translation co-variant (feature that changes under translation) features. Thus, the localization branch may force in learning more translation co-variant features, which can decrease the performance of the box head (category score and box coordinate). To overcome this mismatch of goals, it uses a Prototypical Calibration Block (PCB) to increase the classification accuracy. PCB eliminates high false positive and gives weight to low-score missing sample. This is because, PCB is the prototypical few-shot classifier which is pre-trained on ImageNet as show in figure 5 (Bottom right). PCB takes support and query images and finds the cosine similarity to give a pairwise score, which boosts the box classifier score. Moreover, PCB is used offline and doesn't require any training and therefore can be used directly. The final classification score of the model is a weighted average of the score from the box classifier from the detector and PCB.

It is important to note that DeFRCN jointly trains the entire detector (Faster RCNN) with novel GDL. It uses a standard transfer learning technique to first fine-tuned on a large base set and then again fine-tuned on a novel support set. This setting leads to DeFRCN being simple and outperforming other meta-based and fine-tune-based approaches [29].

This research utilizes algorithm based approach, i.e, DeFRCN because: (1) It achieves the best mAP$_{50}$ of 66.5 among the category (2) FADI needs to perform a redundant step of association which may not useful as the base class and novel class (in our context Japanese Knotweed) are not similar (3) DeFRCN by using GDL allows cross-domain transfer, i.e, the first fine-tuning on base class domain is entirely different from second fine-tuning novel domain. During the novel fine-tuning, RPN is forced to learn to give proposal based on novel class by using a stopping gradient (assisted by GDL) from RPN to backbone. This doesn't allow backbone to get updated by it. The update of the backbone will only be due to RCNN, which is responsible for classification. (3) In Meta-DETR, no performance gain is observed when training support set is increase to more than 10 images (4) DeFRCN by using a few-shot classification block i.e, PCB tries eliminating the high false-positives and increases generalization ability of the model.

III. Methods and Materials

A. Dataset Creation

For creating a diverse dataset of Japanese Knotweed present beside the road, videos were recorded along the Netherlands state road. In particular, a total of 10 sites were visited as shown in the figure 6 : (1) UTrack at the University of Twente (2) Horstlindelaan (3) Wiedicksbeekweg (4) Auke Vleerstraat (5) Strootsweg (6) Sleutelweg (7) Gronausestraat (8) Eschmarke (9) Glanerveldweg (10) Noord Esmarkerrondweg

The following strategies were used for capturing the Japanese Knotweed images:

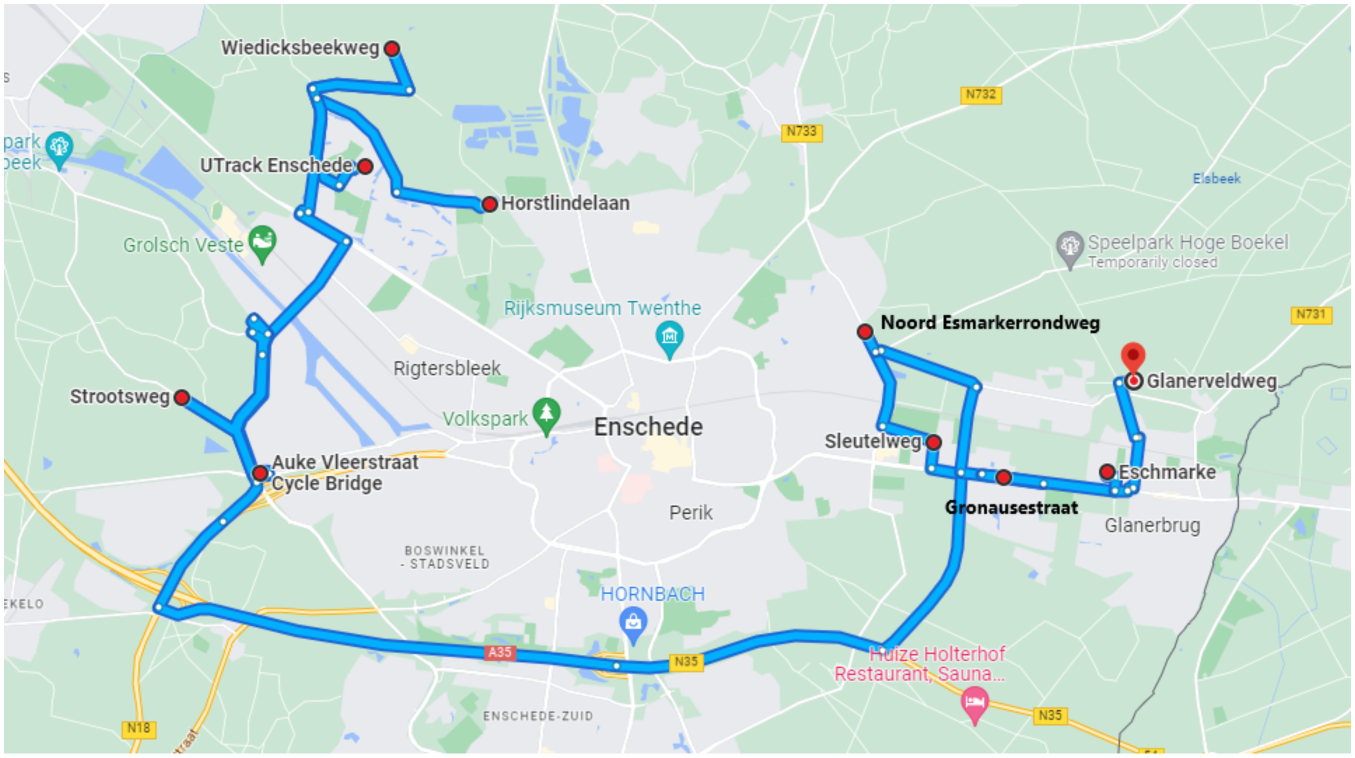


Fig. 6: Red dots representing 10 locations of Japanese Knotweed patch in the stretch of approximately 30 km

1) A mobile camera and a GoPro Hero Plus were mounted on the non-electric bicycle for capturing videos while cycling at average speed of 8-12 km/hr, depending upon the terrain.

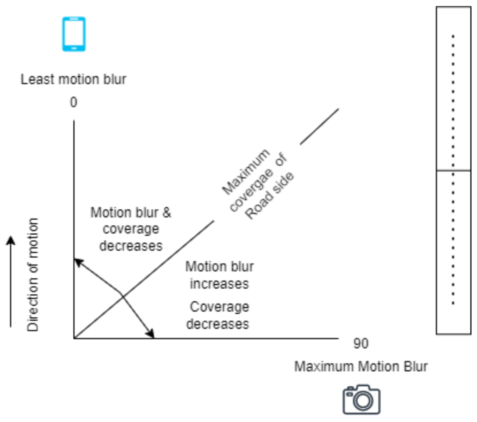


Fig. 7: Camera mounted on bike, depicting motion blur and roadside coverage

2) The mobile camera was mounted at an angle between 0 &45 degrees to the direction of motion. This gives less motion blur compare to angle between 45 & 90 degrees. At 45 degree, a maximum coverage of roadside and moderate motion blur can be obtained, but as mounting is prone to human error, it was ensured that camera is between 0 and 45 during different visit. But, it exerts more demand on subsequent image processing as it is not only invariant to the plant growth stage but also invariant to a different location in the image [8].

3) The GoPro was mounted at 90 degrees to the direction of motion. This angle avoids variation in the size of the plant depending on its location in the image. But, it gives the least coverage and maximum motion blur because change in pixel content during the exposure will be greatest. To decrease motion blur, short exposure setting can be used [8]. However, this research focuses on realistic dataset creation. The figure 7 shows camera mounting, motion blur and roadside coverage.

4) The sites were visited on a biweekly basis on October and November, as shown in figure 2. The same sites were visited multiple times in different environmental conditions. Thus, the dataset has videos of Japanese Knotweed in sunny, cloudy, and rainy weather conditions.

5) The varying distance of Japanese Knotweed to camera is automatically ensured as the plant located beside the road varies at different locations from 1.5 to 12 meters.

6) The dataset was created to include Japanese Knotweed in various stages of the phenological cycle, but due to the plant's growing towards dormant phase, there are more images of the Japanese Knotweed in late growth phase in the dataset.

7) It is also ensured that the dataset contains Japanese Knotweed in patches of different sizes with heterogeneous density.

8) To make the dataset more generic, pictures at random angles (0-45 degree) while a mobile camera still mounted on a bicycle were taken.

Various publicly available database are examined to increase the size of the dataset further. Unfortunately, a popular database such as PlantVillage [30] has images of crop plant leaves. GBIF [31], INatuarlist [32] has Japanese Knotweed images but is biased with scaled up images which is more suited for classification task. Moreover, it does not reflect the complexity of detecting plants from a roadside setting. To solve this problem, the studies suggested using Google Street View images [33]. The primary examination of some locations has shown a promising result. However, it is not used in this research.

B. Cleaning, annotating & organizing dataset

The raw dataset size is approximately 29GB, which only has Japanese Knotweed. The dataset also has videos and photos without Japanese Knotweed which is around 2GB. The dataset mostly have videos(60/30FPS) and thus, many image frames are redundant. Therefore, only 5 image are extracted per second. This decreases the manual labor of examining a massive number of images. Further, frames without, blurred and treated Japanese knotweed pictures were removed. Additionally, the sparse Japanese knotweed image in which cluster are not recognizable has also been removed. The first row in the figure 8 shows a few removed images.

The image were annotated using CVAT [34] and labelled as Japanese Knotweed. As the plants have irregular shape, polygon annotation is an optimal choice. But, the existing DeFRCN architecture doesn't support it. This is verified by analyzing the dataset mapper of DeFRCN GitHub repository. Thus, the following technique has been employed for bounding box annotation:

1) When the camera is positioned at a 0-degree angle to the direction of motion and the plant is in proximity to the camera, multiple bounding box annotations are created to encompass the long stretch of the plant. This is because a single bounding box would not be tight and would incorporate several regions of background comprising other plants.
2) If the plants are far away and angle is between 0 and 45 degree, a single bounding box sufficiently cover the single cluster of the plant.
3) At 90-degree, a single bounding box can cover the entire stretch of the plant.
4) While annotating, in case where branch goes out of the bounding box, it was ensured that majority of the shrub are covered.

In total, 290 bounding box were annotated in 110 images, the representation of which is shown in the second and third row of figure 8. The description of the dataset with respect to source, resolution, angle, and phenology is highlighted in the table I. The distance and plant density is not consider due to significant error in approximation. This answer research question 1 concerning the strategy used to create, and annotate, Japanese Knotweed dataset.

The dataset was divided into a training, validation, and test set with 80% (88 Images), 10% (11 Images), 10% (11 Images). For performing experiments, random shots of 1, 2, 3, 5, 10 are generated from the train set in accordance with PASCAL VOC few shot setting. For example, 1 shot essentially means randomly using a single image with only one instance from the train set for training the models. A total of 30 seeds is created for each shot. An example of how the seed 0 looks is given in the appendix A. Moreover, the complete train set (88 shots with all (223) instances) is also used with all the object detection models.

C. Data Augmentation

Data augmentation is the process of increasing data size by creating more data (images) from the existing dataset. It has been found that a small dataset often leads the model to overfit. Thus, data augmentation makes simple alterations in the existing visual data by applying various geometric and intensity augmentation techniques. This reduces the chances of over-fitting and also increases the model's generalizing ability to the new unseen data [35].

All the object detection model uses online data augmentation. Online data augmentation applies random transformation to the input image during training. In our context, it is preferred over offline data augmentation because of multiple experiment (1, 2, 3, 5, 10) with multiple seeds and different images.

The important geometric transformation for Japanese Knotweed detection includes scaling, rotation, and horizontal flipping. For example, the distance between the plant and the camera changes in a different location. This can be addressed with the scaling technique, which adds examples of such variations in the dataset for model training. Similarly, the mounting of a camera is prone to human error, random rotation between 0 & 45 degrees can model different pose of the plants. The intensity transformation includes contrast, saturation, brightness, and color enhancement technique. These aim to model color variation that arises due to environmental conditions and different time of a day. All the selected model incorporates these augmentation techniques, which is further elaborated with respect to the object detection models in the section V.

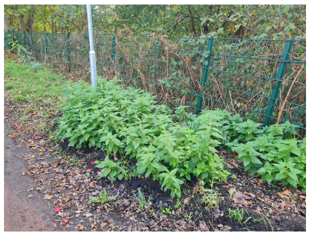
IV. Evaluation metric

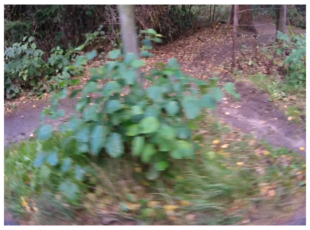
The performance of the model can be measured with the help of evaluation metrics. It can be used to evaluate a model's quality or to assess how well the various models work to determine the best one. Mean Average Precision or mAP_{50} with IOU (Intersection Over Union) 0.5 has been consider under this research as it is standard in case of few-shot learning methods.

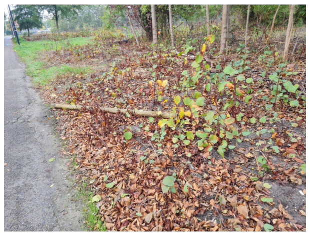
TABLE I: Japanese Knotweed Dataset Description

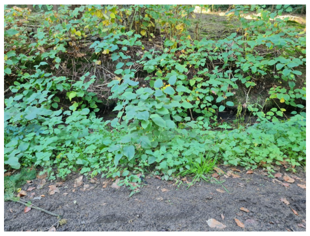
Source	Resolution	Angle (degree)	Phenology$^{\alpha}$	Number of Images (110)
GoPro	1920x1080	90	Late-growth	9
Mobile	1920x1080	0-45	flower	3
Mobile	1920x1080	0-45	Early-growth	26
Mobile	1920x1080	0-45	Late-growth	20
Mobile	1920x1080	0-45	Dying	28
Mobile (Picture)	4032x3024	90	Early-growth	1
Mobile (Picture)	4032x3024	90	flower	1
Mobile (Picture)	4032x3024	90	Late-growth	12
Mobile (Picture)	4032x3024	90	Dying	1
Mobile (Picture)	4032x3024	0-45	Early-growth	7
Mobile (Picture)	4032x3024	0-45	Flower	1
Mobile (Picture)	4032x3024	0-45	Late-growth	1

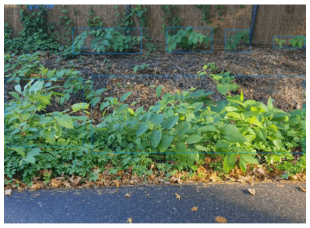
$^{\alpha}$Quantification is approximate on visual assessment and may have some error, *The inclusion of Japanese Knotweed's distance and density has been omitted due to significant errors in approximation.

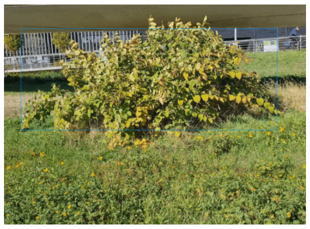

(a) No JK

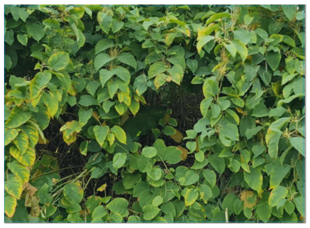

(b) Blurred JK

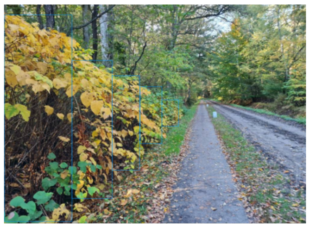

(c) Treated JK

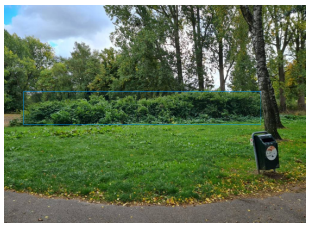

(d) JK cluster not recognisable

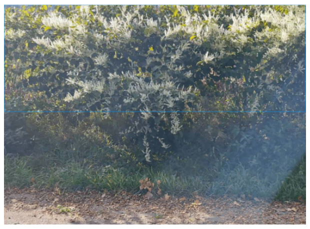

(e) 90°, EG, $<$ 2 meter

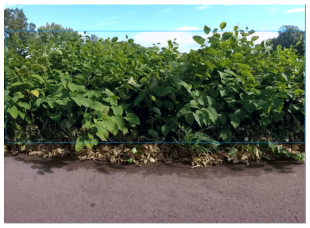

(f) 90°, LG, $>$ 5 meter


(g) 90°, LG $<$ 1 meter


(h) 0°, Dying


(i) 0°, LG, $>$ 10 meter


(j) Illuminated, flower


(k) GoPro JK

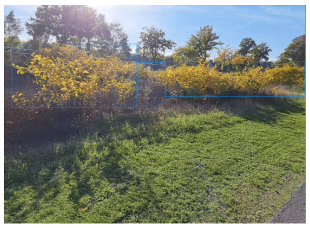

(l) JK at random angle

Fig. 8: Annotated images representing the Japanese Knotweed (JK) dataset variability. EG, LG denotes early and late growth respectively. Distances are approximate.

The precision, recall and IoU are consider while calculating the average precision. So, understanding its definition is crucial.

A Precision is defined as:

$$P = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} = \frac{\mathrm{TP}}{\text{all detections}} \tag{2}$$

A Precision gives a percentage of correct prediction over all the predictions. A recall is defined as:

$$R = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} = \frac{\mathrm{TP}}{\text{all ground truths}} \tag{3}$$

Recall gives a percentage of correct prediction over actual cases.

An IoU (Intersection Over Union) is the ratio of the overlapping of ground truth and predicted area to the total area. The IoU close to 1 implies that the model perfectly anticipated the object. On the contrary, an IoU close to 0 implies that the model didnt predict object coordinates. Thus, a threshold is always defined. For example, IOU with a threshold set to 0.5 will detect an object as true positive if the overlap between the predicted bounding box and the ground truth bounding box is more than 0.5. If the overlap is less than 0.5, it means it is a true negative.

The average precision is an area under precision and recall curve. The high area under the curve indicates both precision and recall are high. The area is generally a zigzag curve, and hence we use 2 methods to approximate it. This research uses 11 point interpolation method. In the 11-point interpolation, the shape curve is summarized by averaging the maximum precision value at a set of 11 equally spaced recall values [0.1, 0.2, ... , 1]. Mathematically, it is given by:

$$\mathrm{AP11} = \frac{1}{11} \sum_{R \in \{0, 0.1, \ldots, 0.9, 1\}} P_{\text{interp}}(R) \tag{4}$$

where

$$P_{\text{interp}}(R) = \max_{\tilde{R}:\tilde{R} \geq R} P(\tilde{R})$$

Here, instead of using the precision $P(R)$ observed at each recall level R, the AP is obtained by considering the maximum precision $P_{\text{interp}}(R)$ whose recall value is greater than R [36].

V. Experimental Setup

A. Faster RCNN

The research uses a Faster RCNN model from detectron2 model zoo. Detectron2 is an open-source object detection and segmentation framework built on top of PyTorch. It is developed by Facebook AI Research [37]. Detectron2 is selected over TensorFlow and PyTorch because: 1) It has intuitive API which makes building of object detection model simple. 2) It is highly flexible as it provides a modular architecture to mix and match different components for building a model. For instance, it provides various pre-trained backbone networks, such as ResNet or EfficientNet, to extract features from the input image. 3) It provides the highest performance, mAP_{50} of 41 on the coco dataset.

The Faster RCNN model has been configured to have a similar base architecture as utilized by DeFRCN. Thus, in our implementation, it uses ResNet-101 as a backbone for extracting the feature from input images. The ROI Head uses Res5 and C4 features to incorporate both high-level semantic features and low-level spatial features for predicting bounding boxes with class labels and confidence scores. The final classification layer was adapted to match the Japanese Knotweed class.

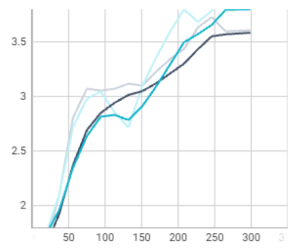
Detectorn2 recommends resizing the image (height & width) within the range of [800, 1333] to achieve the best performance. Therefore, the images and annotations are resized to their default image setting, which is 1333x800 pixels. ImageNet pixel mean and standard deviation has been used for normalizing the image, as it provided better performance.

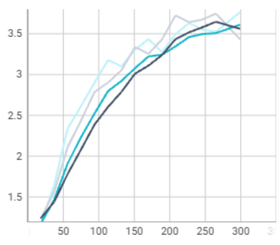
Seven different online data augmentation has been employed while training the model. These are: (1) Random change in brightness, contrast, and saturation between 0.8 to 1.2 [38], (2) Random horizontal flip, (3) Random lighting by 0.7 (changes brightness and illumination), (4) Random rotation between 0 and 45 degrees (5) Random resizing of the image between image's max (1333) and min (800) dimension (for exposing the network to different scales of objects during training). The parameter values are carefully selected to avoid unwanted and unrealistic effects. Few examples are shown in the figure 10.

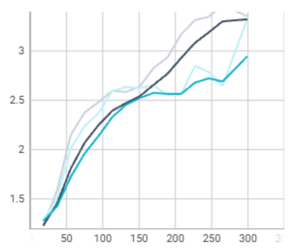
The batch size of 2 is used with all experiments. After every 20 iterations, the network was evaluated on the validation set. Additionally, the learning rate has been kept the same, i.e., 0.001 for all the experiments. The iteration for 1, 2, and 3 shot are 300, for 5 shot it is 500, and for 10 shot it is 590. These parameters are the same for experiments concerning with and without data augmentation, except in 88 shot where iterations are 300 and 550 for without and with data augmentation.

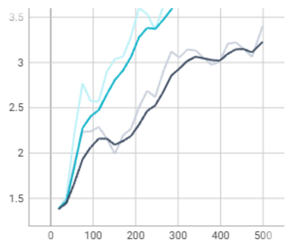
A pre-trained weight, on ImageNet, is used to initialize the model. Further, transfer learning and fine-tuning on custom Japanese Knotweed class is performed. In general, fine-tuning the entire network is only used when the target dataset is large enough to train parameters of a network. Thus, the number of stages to freeze depends on the amount of target training dataset [39]. Moreover, a transfer learning is more effective if the source and target domain are similar, however, for comparison with few-shot algorithm which allows cross domain transfer, ImageNet weight was given priority.

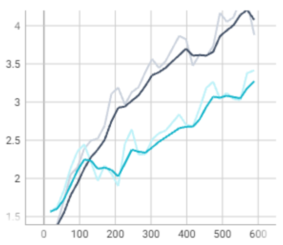
In our context, the target dataset is small with a maximum of 88 images, and other experiments have only 1, 2, 3, 5, 10 images for training. Therefore, freezing the initial layers of backbone (conv1, conv2_x, and conv3_x) may serve as a feature extractor as these layers typically learn low-level features such as edges and color, which do not change significantly across the visual recognition tasks [40]. Higher layers (conv4_x and conv5_x) responsible for high-level features,

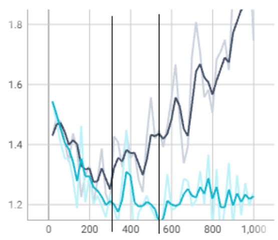

(a) 1 shot val curve

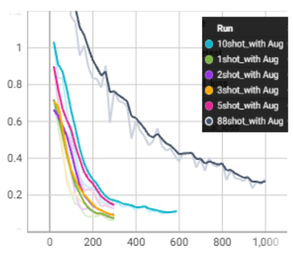

(b) 2 shot val curve


(c) 3 shot val curve

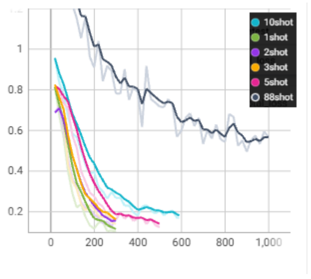

(d) 5 shot val curve


(e) 10 shot val curve


(f) 88 shot val curve



(g) Loss curve with aug.



(h) Loss curve without aug.

Fig. 9: The figure depicts the validation plot for 1, 2, 3, 5, 10 (seed 0 as an example) and 88 shots (complete dataset) with (blue line) and without data augmentation (black line). It also gives loss curves for all experiments. The transparent lines and the bold line denote the actual plot and the smooth plot. The vertical line in 88 shot denotes the iteration at which model weights are collected, which are 300 and 550 for without and with data augmentation, respectively.

which are specific to the dataset, i.e, are fine-tuned.

Figure 9 represents the validation plot and loss plot for experiments 1, 2, 3, 5, 10 (on seed 0 as an example) and 88 shots with and without data augmentation. We can see that the validation curve rises sharply from the beginning in experiments 1, 2, 3, 5, and 10 for both cases (with and without data augmentation) and, the loss curve decreases smoothly. This indicates that the model is overfitting on the validation set. On the contrary, the validation curve for 88 shots with and without data augmentation continues to decrease till 300 and 550 respectively. It is only after 300 iterations, the validation curve starts to increase for without augmentation. With data augmentation, the validation curve becomes constant after 550 iterations, indicating it is limiting overfitting. This shows that the model can generalize to the unseen image if the complete dataset is given. Thus, model weights are collected to verify this. However, the model with data augmentation is also not robust, which can be seen in figure 14. The first row of the figure represents the model prediction on classes other than plant class. Here, 25 images were given for prediction of which, the model falsely labeled 23 images with Japanese Knotweed. The second row represents the model prediction on other plant classes with 13 images, of which 12 images were labeled as Japanese Knotweed. Finally, the last row has images of Japanese Knotweed. Here, 11 images with 37 instances are used, for which mAP$_{50}$ was 35.17 with data augmentation. But, the prediction has a high false positive. This can also be verified from the figure, where trees and other plants in the image are labeled as Japanese Knotweed. The model gives biased prediction because the model has only been trained on one class and weights are optimized to minimize loss for Japanese Knotweed, so it doesn't know how to distinguish positive and negative samples.

Therefore, we can conclude Faster RCNN model overfits for experiment 1, 2, 3, 5, 10 and for experiment 88 it doesn't to give a robust prediction in detecting Japanese Knotweed when trained on one class. It has high false positives, where all other classes are labelled as Japanese Knotweed. This answers research question '2a' relating to Faster RCNN performance.

B. DeFRCN

This research utilizes and extends the implementation of DeFRCN [41]. The DeFRCN repository inherits from FsDet which is also based on the detectron2 framework. As mentioned above, DeFRCN extends Faster RCNN with GDL and PCB. DeFRCN has the same ResNet-101 as the backbone, with Res5 and C4 as ROI Head. However, unlike Faster RCNN where fine-tuning is performed only once, a two-step fine-tuning procedure with the network being initialized with pre-trained ImageNet weights is performed. First, it is fine-tuned on an abundant base class comprising 15 categories from the PASCAL VOC dataset. Then, it is again fine-tuned with the Japanese Knotweed novel class. Similar to Faster RCNN, the fine-tuning has been performed for stages 4 and 5 of ResNet101.

The data augmentation procedure also remains the same as discussed under Faster RCNN. Figure 10 shows a few examples, particularly rotation, and random flip. Furthermore, the parameter value which contributes to the unnatural augmented images is discarded. Additionally, for normalization, ImageNet pixel means and the standard deviation are considered which is again similar to Faster RNN because it provided better mAP$_{50}$ on the validation set.

The following modifications & extensions have been made to the existing model:

1) Random seed generator for creating 30 seeds, each with 1, 2, 3, 5, and 10 random non-repeated images (shots). The script uses one object instance from each image. For 88 shots, all instances have been used.
2) A data loader for seeds along with its metadata.
3) A custom trainer with a validation hook to perform additional evaluation and logging for the validation set.
4) A custom datamapper for online data augmentation with manual seed. This ensures the same augmentation is applied across multiple experiments.
5) A visualization script for the dataset.
6) A configuration file for considering custom 88 shots.
7) Various modifications have been performed to adapt the DeFRCN for Japanese Knotweed detection such as multiple image instance usage for 88 shot, custom test set registration etc.

The details and relevant script snippets for each of the steps have been provided in the appendix A.

The network training procedure of DeFRCN differs from the meta-learning method, where episodic training is used (PCB is trained based on meta-learning, but DeFRCN is using it only during inference). In this case, it generally uses the TFA approach [42] for training and obtaining the results from the model. It creates multiple support sets (30 seeds) for each experiment, and an average mAP$_{50}$ on the test set is obtained from it. For instance, for 1 shot, the model is trained on 1 random different image 30 times. This gives 30 networks. Then, the mAP$_{50}$ on test set for 1 shot is based on the mean average of 30 such networks. The learning rate is, 1xe^{-4} for all the networks under same and across experiments. The iteration for the 1, 2, 3, experiment is 5000, for 5, 10 and 88 shots are 6000. The parameters are the same for with and without data augmentation. The batch size is 2 for all experiments. Additionally, the network is trained on varying image sizes ranging from 480 to 800 resolution with 32 pixels.

Shots	Without Augmentation	With Augmentation
1	3.22	5.33
2	5.43	4.83
3	5.41	6.45
5	8.03	7.31
10	10.10	8.30
88	21.9	27.1

TABLE II: mAP$_{50}$^{test} for 1, 2, 3, 5, 10 and 88 shots.

Table II summarize the mAP$_{50}$ on test set obtained from all the experiments over 30 seeds with and without data augmentation. There is only one seed for 88 shots, as the complete train set is used. It can be observed from the table that as the number of shots increases, the few-shot model gives better performance. Furthermore, data augmentation should be used judiciously with an extreme shortage of training examples, as indicated by experiments 2, 5 and 10, where a decrease in performance was observed, and experiments 1 and 3, where an increase was observed. However, when the dataset is increased to 88 shots, a significant performance improvement of 5.21 mAP$_{50}$ is achieved. Moreover. DeFRCN avoids overfitting, which can be seen from the validation plots for seed 0 given in the appendix in figure 53, 52, 51, 50, 49.

The common setting in the few-shot model during inference is to give support and the query image. But, DeFRCN doesn't explicitly require support images, and instead, uses the train image given to the detector (Faster RCNN) for PCB. For instance, 1 shot only uses 1 support image. Thus, a study was conducted to understand the model performance by increasing the size of the support set. By exposing PCB with more images during inference, a better prototype with more generic and diverse features of the Japanese Knotweed can be created. This may increase the performance from PCB as the classification score in DeFRCN is a weighted sum of classification score from detector (Faster RCNN) and PCB as mentioned in the below equation:

$$C = 0.5 \cdot C_{detector} + 0.5 \cdot C_{pcb} \tag{5}$$

The table III summarizes the results obtained from seed 0 for different experiments. We can observe that there is a sharp increase in the mAP$_{50}$. However, the overall performance of the model declined further when it examined over 30 seeds. This can be verified from table IV.

Shots	Limited Support Set	Large Support Set
1	2.32	4.54
2	2.76	13.63
3	5.54	15.90
5	9.85	20.10
10	13.05	23.47

TABLE III: mAP$_{50}$^{test} on seed 0 for 1, 2, 3, 5, 10 with limited and large support set of 185 instances of Japanese Knotweed during inference with data augmentation.

Shots	Limited Support Set	Large Support Set
1	5.33	3.31
2	4.83	3.94
3	6.45	5.44
5	7.31	7.89
10	8.30	6.25

TABLE IV: mAP$_{50}$^{test} for 1, 2, 3, 5, 10 with limited and large support set of 30 seeds.

Figure 18 visualizes the prediction by 1, 2, 3, 5, and 10 shot models from seed 0 as an example. The weights of data augmentation with a large support set were selected as

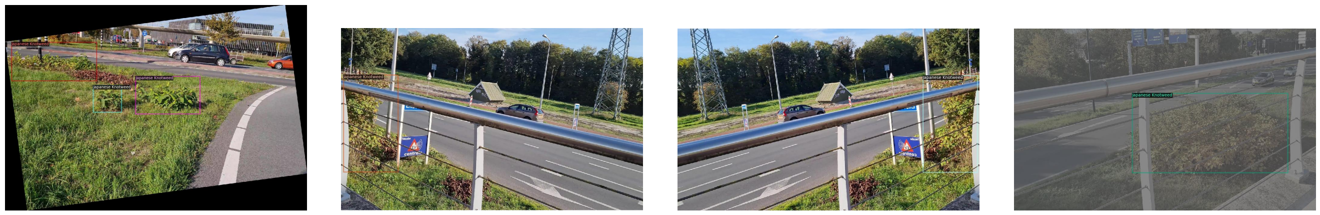


Fig. 10: The figure depicts a few of the data augmentation applied to the image. The first image represents rotation. The second & third image shows horizontal flip. Finally, the fourth image shows a discarded, unrealistic image of Japanese Knotweed.

performance improvement was observed on seed 0. The first row represents ground truth for all the images. It can be seen from the last row which represents 10 shot prediction, with increasing number of shots more instances of Japanese Knotweed are detected. Also, the model tries to reduce false positive (3rd row, 3rd image). However, there are missing instances of Japanese Knotweed which can be seen from 6th row, 1st image when compare to ground truth. Additionally, with 5 shot in row 5, for the image 3, there is no prediction when compared to 3 shot and 10 shot. It can also be seen when less image are available for training (1 shot) model is not able to detect instances of Japanese Knotweed (row2 image 1 and 3).

Figure 15 visualizes the prediction by the 88-shot model. The weights from the data augmentation model have been used. The first row of the figure represents the model prediction on classes other than plant class. Here, 25 images were used for testing, of which, the model falsely labeled 5 images with Japanese Knotweed. The second row represents the model prediction on other plant classes with 13 images, of which 7 images were labeled as Japanese Knotweed. Finally, the last row has images of Japanese Knotweed. Here, 11 images with 37 instances are used, for which mAP_{50} was 27.1. In the last row, image 1, although the model correctly identifies it as Japanese knotweed, there is a localization error with 2 bounding box instead of one. In image 2, there is a localization error and missed detection. Image 3 and image 4, it correctly detects Japanese knotweed. This can be verified with the ground truth image shown in the figure 17.

There is a vast improvement in detection compared to Faster RCNN because of fewer false positives, which makes DeFRCN more robust. This answers research question '2b' concerning DeFRCN performance, as the finding above suggested it is more robust compared to Faster RCNN and can detect Japanese Knotweed in a limited dataset. Additionally, a study with data augmentation suggested that a mAP_{50} increase of 19.1% in the case of 88 shots, while it should be used cautiously when the number of images is extremely limited.

Furthermore, to gain a deeper understanding of the DeFRCN architecture, an ablation, and resolution study was conducted. This study aimed to analyze the impact of GDL and PCB on the performance of the architecture and to investigate how changes in image resolution affect the accuracy of the model.

1) Ablation study: Following the DeFRCN research paper, this research also uses 10 shots but on seed 0 for the ablation study. Unlike the research paper, a large support set during inference has been used, as an increase in performance was observed for seed 0. The impact of GDL and PCB in the Faster RCNN architecture for detecting the Japanese Knotweed novel class can be seen from the table V. It can be seen that PCB gives a major boost in increasing the performance of DeFRCN. When combined with GDL, it further enhances it.

FRCN	GDL-N	PCB	JK (WA)	JK (NA)
✓			-	-
✓	✓		8.03	5.96
✓		✓	18.18	13.5
✓	✓	✓	23.04	18.18

TABLE V: $\text{mAP}_{50}^{\text{test}}$ of different components in DeFRCN architecture with data augmentation (WA) and without data augmentation(NA) for 10 shots seed 0 and large support set.

2) Resolution vs mAP_{50}^{test}: For studying resolution impact in mAP_{50}, seed 0 to seed 29 for 10 shot with an interval of 5 seeds are selected. At each image resolution(480x480, 608x608, 736x736, 864x864, 992x992, 1120x1120, 1248x1248) corresponding mAP_{50} was recorded at each seed and a box plot was constructed for 10 shots. Thus, for instance, at 480x480, mAP_{50} from seeds 0, 4, 9, 14, 19, 24, and 29 are used. The findings are presented in figure 11. We can see that at higher resolutions (992, 1120, and 1248), the range is short, indicating a consistent performance at different seeds which indicate less variance. For instance, at resolution 480 at different seeds, we can observe drastic changes in mAP_{50} indicating high variance, which is not the case with higher resolution. Additionally, the Inter Quartile Range (IQR) is also short in higher resolution, indicating out of 7 seeds at least 50% of the seeds will have a similar mAP_{50}. The lower resolution has a longer IQR, indicating a large difference in mAP_{50} of 50% seeds. Moreover, a performance gain is observed with higher resolution.

C. YOLOv7

The PyTorch implementation of the YOLOv7 has been used in this research [43]. As mentioned earlier, YOLOv7 has six different variants. This research uses the standard YOLOv7 model, which provides an $\text{AP}_{50}^{\text{test}}$ of 69.7% for benchmarking.

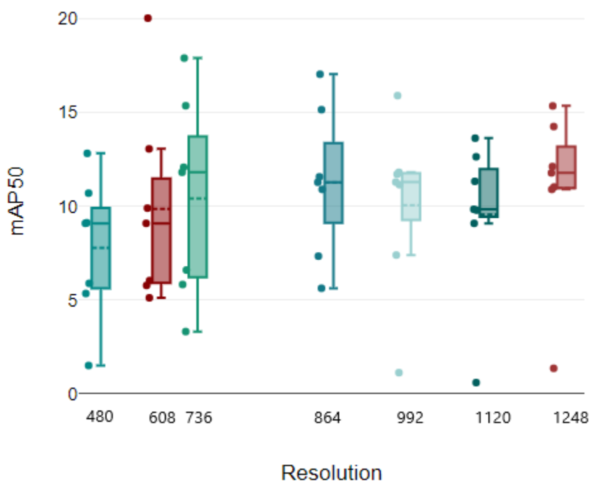


Fig. 11: mAP_{50} vs resolution box plot using 10 shot with seed interval of 5 with data augmentation and limited support set. The resolution has a step of 128 pixels ranging from 480 to 1248.

The annotations were first updated from Pascal VOC XML format (used by Faster RCNN and DeFRCN) to YOLO text file format (Object Class, X Center, Y Center, Width, Height). This is because CVAT doesn't support exporting to YOLOv7 format. Further, the standard image size of 640x640 resolution has been used followed by normalizing the image and bounding box. The default hyperparameter file was modified with a learning rate (lr0) of 0.001 and rotation between 0 & 45. The learning rate was selected based on performance on the validation set (box loss at further lower learning rate started to increase). Data augmentation-specific parameters such as HSV (Hue adds variations of color in the image, Saturation, and Value for altering lighting conditions), scale, translate, and left-right flip remained the same. YOLOv7 provides mosaic augmentation (takes four images and forms a composite) for learning different contextual information around an object, which improves its generalizing ability of the model. Thus, it is also used along with other data augmentation [44].

A standard transfer learning approach, with the pre-trained YOLOv7 weights on the coco dataset, was used to initialize the model. The half of the backbone (25 layers) is fine-tuned with batch size 8 and epoch 500 for all the experiments.

To understand the model performance, we first need to examine loss computation in YOLOv7 as unlike Faster RCNN which gives a plot for total loss, YOLOv7 gives individual plots:

$$\text{TL} = w_1 \times \text{Obj. Loss} + w_2 \times \text{Class. Loss} + w_3 \times \text{Box Loss} \quad (6)$$

Total loss (TL) is calculated as a weighted sum between objectness loss, classification loss, and box loss. w_1, w_2, and w_3 are user-defined hyperparameters (default parameters in our context with the highest weightage to objectness loss with 0.7). Intuitively, objectness loss measures the accuracy of the model in detecting if an object is present in the predicted bounding box. Classification loss gives the accuracy of model classification of an object inside the predicted bounding box, and box loss measures the inaccuracies in localizing the predicted bounding box.

Technically, the objectness loss is calculated in 4 steps: (a) The YOLO network predicts the probability for each anchor box, depicting if it contains an object. This is called the objectness score. (b) During network training, each anchor box is matched with a ground truth bounding box to obtain a matched anchor box with the highest Complete IoU (CIoU) score. The CIoU score between the matched anchor box and ground truth bounding box is then used as the ground truth. CIoU aims at measuring the distance between the center point of the predicted and ground truth bounding box, along with their aspect ratios. (c) The predicted objectness probability is compared to the ground truth CIoU value using the Binary Cross Entropy Loss (BCEL). (d) To obtain the objectness loss score, the BCEL value is multiplied by a hyperparameter w_1. This hyperparameter controls the weight of the objectness loss in the overall loss function.

Box Loss is calculated as w_2 x mean(1 - CIoU) between all predefined anchor boxes and their matched target (adjusted anchor box while network training).

Classification loss is computed as the BCE between the predicted class probabilities and the corresponding one-hot encoded vector (transform categorical data to numerical vectors). The BCE value is then multiplied by a hyperparameter w_3 to give the final classification score.

Thus, the most important loss is objectness loss, as it prioritizes the presence of an object in the predicted bounding box based on which localization will be performed.

The figure13 depicts train & validation plot for all the experiments. With 1 shot (purple), we can observe that obj val loss vis-a-vis obj train loss continues to decrease. Thus, the model weight is collected at iteration 499 and mAP_{50} was recorded to be less than 0. For experiments 2 (orange), 3 (black), and 5 (pink) the weights are collected at 474, 299, 374, and mAP_{50} was recorded to be less than 0 again. For 10 shots (cyan), weight is collected at 74, as after this iteration the objectness loss increases, indicating overfitting. The mAP_{50} was recorded to be 2.3. Finally, for 88 shots, the object loss increases from the beginning and then continues to decrease till epoch 94 after which it increases again while its corresponding train object loss keeps on decreasing. This could be because during the initial phase of training, the model is learning to recognize to generalize patterns from the data. As a result, the validation loss can sometimes increase because the model is overfitting the training data. Once the model has learned the patterns, the validation loss begins to decrease again. Thus, the weight is collected around 94 epoch and

mAP_{50} of 32.60 was recorded.

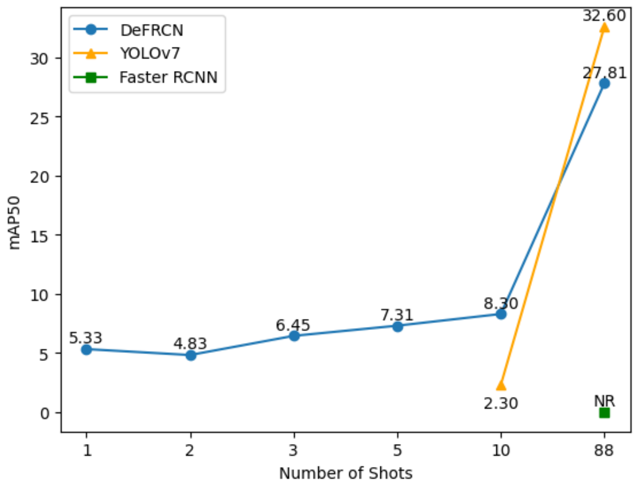


Fig. 12: mAP_{50} for Faster RCNN, DeFRCN and YOLOv7 object detection models under various experiments. NR indicates not robust.

Figure 16 shows prediction from the YOLOv7 model. The first row of the figure represents the model prediction on classes other than plant class. Here, 25 images were used for prediction of which, the model correctly predicts none of the images as Japanese Knotweed. The second row represents the model prediction on other plant classes with 13 images, of which 7 images were labeled as Japanese Knotweed. Finally, the last row has images of Japanese Knotweed. Here, 11 images with 37 instances are used, for which mAP_{50} was 32.60 with data augmentation. In the last row, the model is correctly able to predict Japanese Knotweed as shown in the image 1, 2 and 4. However, the model gives wrong prediction in the 3rd image, where the tree and grass are also identified as Japanese Knotweed compare to the ground truth shown in 17.

Furthermore, it was found that image augmentation plays a major role in defining the performance of the YOLO model. Without augmentation, the model mAP_{50} is 19.9.

Thus, we can conclude that YOLOv7 has no false positives compared to Faster RCNN & DeFRCN in other classes. Moreover, it has the same number of false positives as that of DeFRCN on other plant classes, which is 7. Although it mAP_{50} is nearly zero for the experiments 1, 2, 3, 5, it offers the highest mAP_{50} with complete dataset. This answers research question '2c' concerning the performance of state-of-art object detection methods.

VI. Discussion

Figure 12 shows the mean average precision (mAP_{50}) obtained from three object detection models, namely Faster RCNN, DeFRCN, and YOLOv7 for detecting Japanese Knotweed under limited data scenarios. The results of the experiments show that when trained on a few images (1, 2, 3, 5, and 10 shots), the Faster RCNN model completely overfits. DeFRCN provides 5.33, 4.83, 6.45, 7.31, and 8.30 mAP_{50}. We also observed that as the number of shots increases, more instances of Japanese Knotweed are detected, indicating a rise in true positives. On the contrary, YOLOv7 fails to generate predictions, resulting in nearly 0 mAP_{50} for 1, 2, 3, and 5 shots. For 10 shots 2.3 mAP_{50} was observed.

When the complete dataset is used, the Faster RCNN model doesn't overfit but has poor generalization ability, resulting in extremely high false positives. In contrast, the state-of-the-art YOLOv7 model achieves the highest mAP_{50} of 32.6, even higher than DeFRCN's which gives 27.1. This demonstrates that YOLOv7 can achieve good results even with a low dataset.

Data augmentation plays a crucial role in object detection, and its impact was observed in all three models. For the Faster RCNN model trained on the complete dataset, data augmentation helped reduce the problem of overfitting. In the case of the DeFRCN model, the response to data augmentation was mixed. For instance, there was an increase in performance for shot 1, 3, and 10, but a decrease for shot 2 and 5 indicating data augmentation should be used in caution with limited datasets. However, when trained on the complete dataset, the DeFRCN model showed a sharp rise of 19.18% in mAP_{50}.

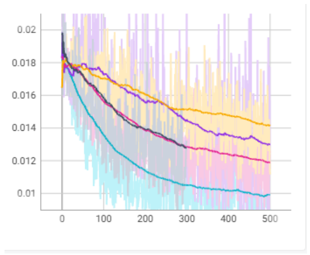
We also analyzed the impact of the support set and resolution concerning DeFRCN. With higher resolution, it was found that model performance increased with a decrease in variance. The support set experiment suggested that under a low dataset of 1, 2, 3, 5, and 10, increasing the support set in PCB doesn't necessarily increase the performance.

VII. Conclusion & Future Scope

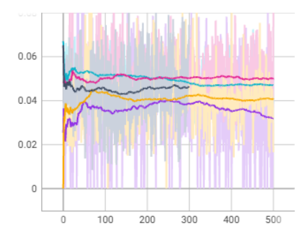
A. Conclusion

The main research question addressed in this study focused on achieving object detection for Japanese Knotweed using a limited annotated dataset while considering various conditions, such as distance, lighting, illumination, angles, resolution, and plant phenological cycle. To answer this question, in the DeFRCN implementation, images with diverse conditions (seeds) and conducted experiments with varying numbers of images (shots) are employed. The formation of seed is random, aimed to simulate real-world scenarios. Specifically, the study formulated experiments with 1, 2, 3, 5, 10 shots with 30 seeds, as well as a complete dataset experiment.

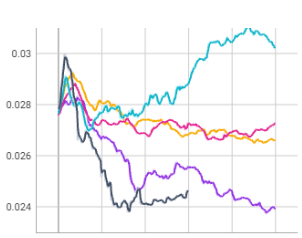
The experiments with 1, 2, 3, 5, and 10 shots demonstrated that the performance improvement of the DeFRCN model was directly proportional to the number of images used. The experiments involving multiple seeds provided insights into the minimum performance achievable by the DeFRCN model. Furthermore, the findings indicated that by carefully selecting appropriate data augmentation techniques and leveraging a large support set, the model's performance could be further increased. Data augmentation particularly showed a significant



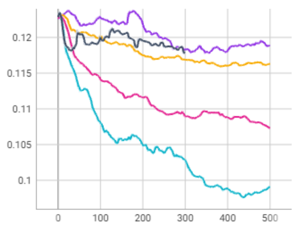
(a) Obj train loss (1,2,3,5,10)



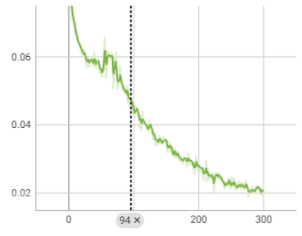
(b) Box train loss (1,2,3,5,10)



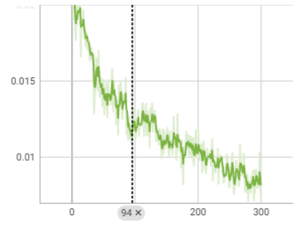
(c) Obj val loss (1,2,3,5,10)



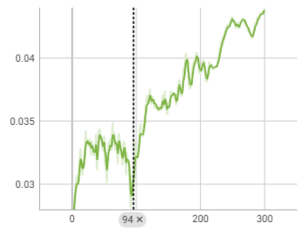
(d) Box val loss (1,2,3,5,10)



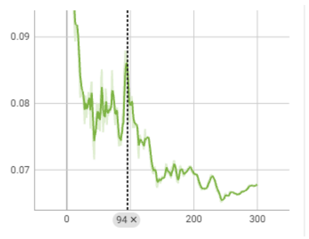
(e) Obj train loss



(f) Box train loss



(g) Obj val loss



(h) Box val loss

Fig. 13: The figure depicts train & validation plot for 1(purple), 2(orange), 3(black), 5(pink), 10(cyan) on seed 0 and 88 shots (green). The transparent lines and the bold line denote the actual plot and the smooth plot. The vertical line in 88 shot denotes the epoch at which models weights are collected.

increase of 19.1% when applied to the complete dataset experiment.

A comparative analysis was conducted between the DeFRCN model and a generic object detection model trained on a complete dataset. The results highlighted that the Faster RCNN model, when trained on a single class and exposed to other classes, struggled to distinguish between classes, leading to a high number of false positives. In contrast, the YOLOv7 model avoided confusion and provided accurate predictions, giving the highest performance of mAP_{50} among the models. The DeFRCN model, by using PCB and aggregating the classification score between PCB and the detector, successfully reduced false positive detections, contributing to its improved performance.

B. Future Scope

1) Study on the few-shot model deployment can be performed. Currently, it is challenging as detectron2 no longer provides ONNX conversion support directly. This is explained in brief in the Appendix.
2) Currently, the research study relies on the GDL's ability to perform a cross-domain transfer of knowledge from a base class. The study can be conducted to increase the knowledge transfer from a similar large base class (Plant class) which may have the potential to improve the accuracy further.

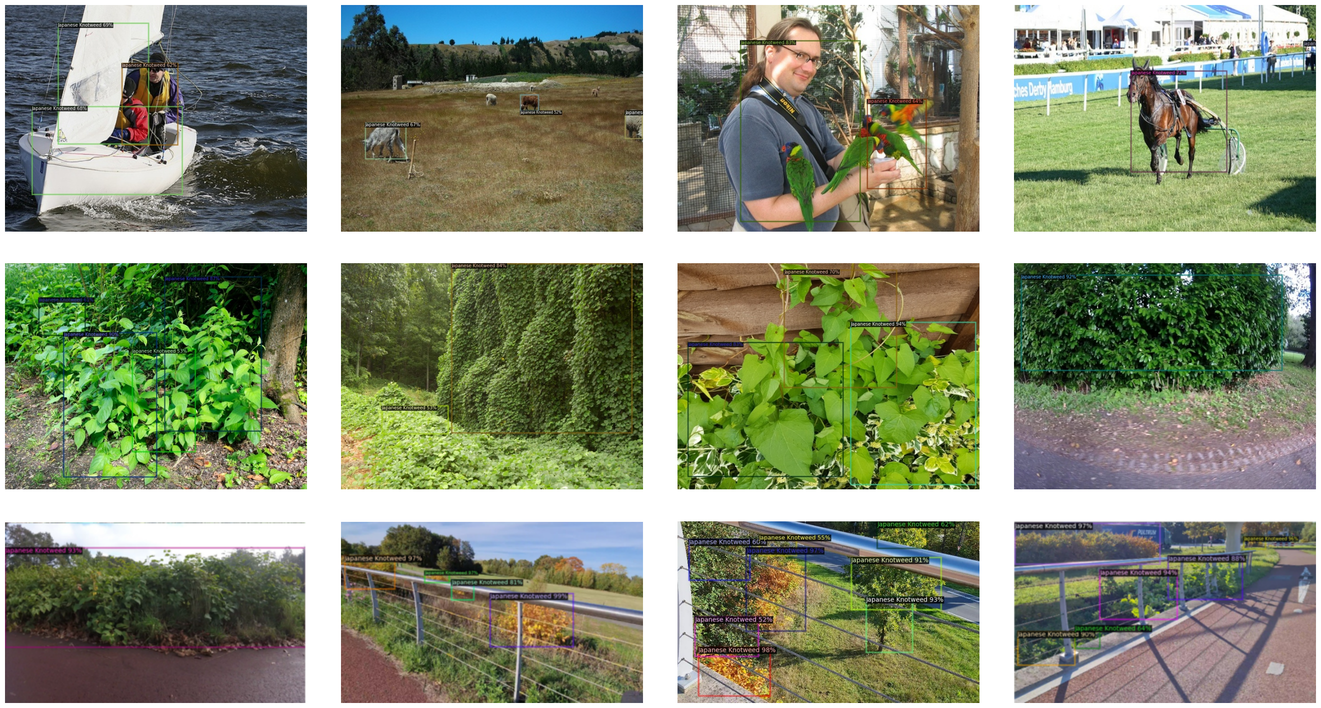


Fig. 14: The figure shows the robustness of 88 shot model to other classes, different plant species and its generalizing ability on the test set comprising Japanese Knotweed using Faster RCNN with data augmentation.

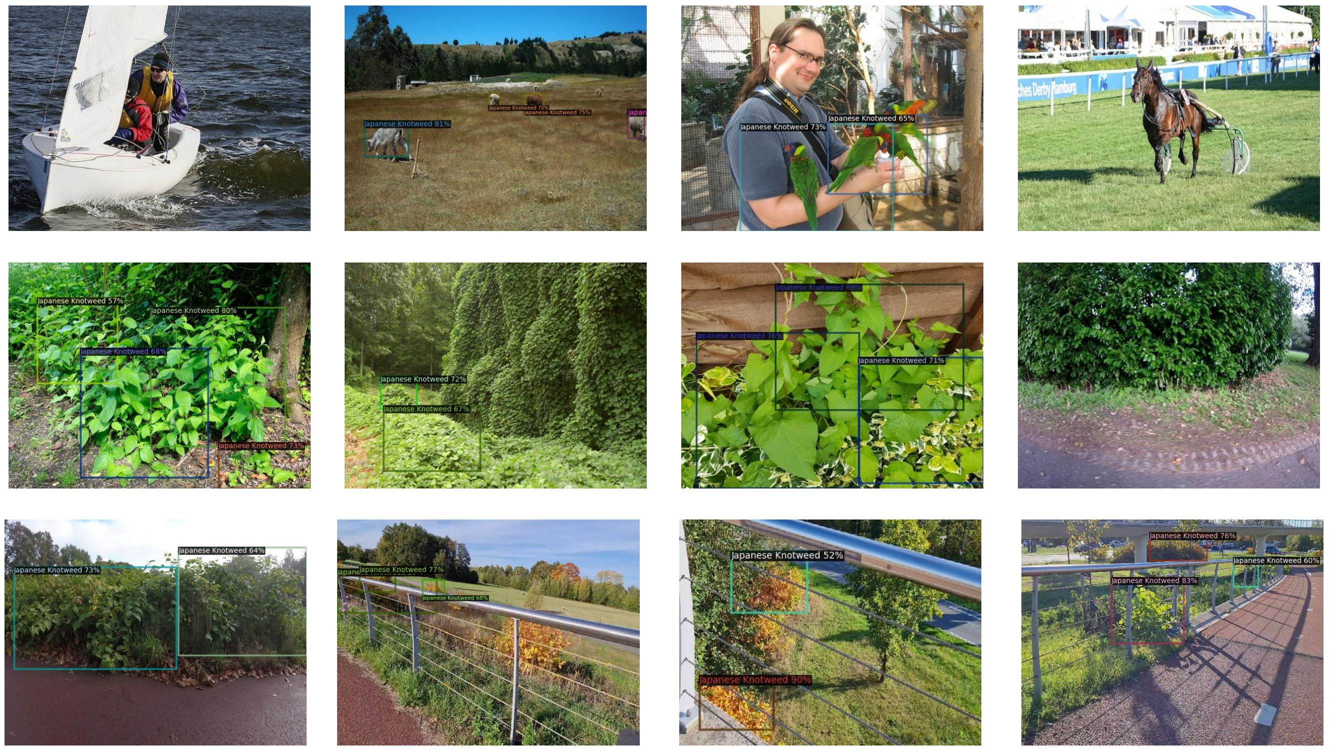


Fig. 15: The figure shows how the 88 shot model with data augmentation generalizes to other classes, different plant species and its predictions on the test set comprising Japanese Knotweed using DeFRCN model.

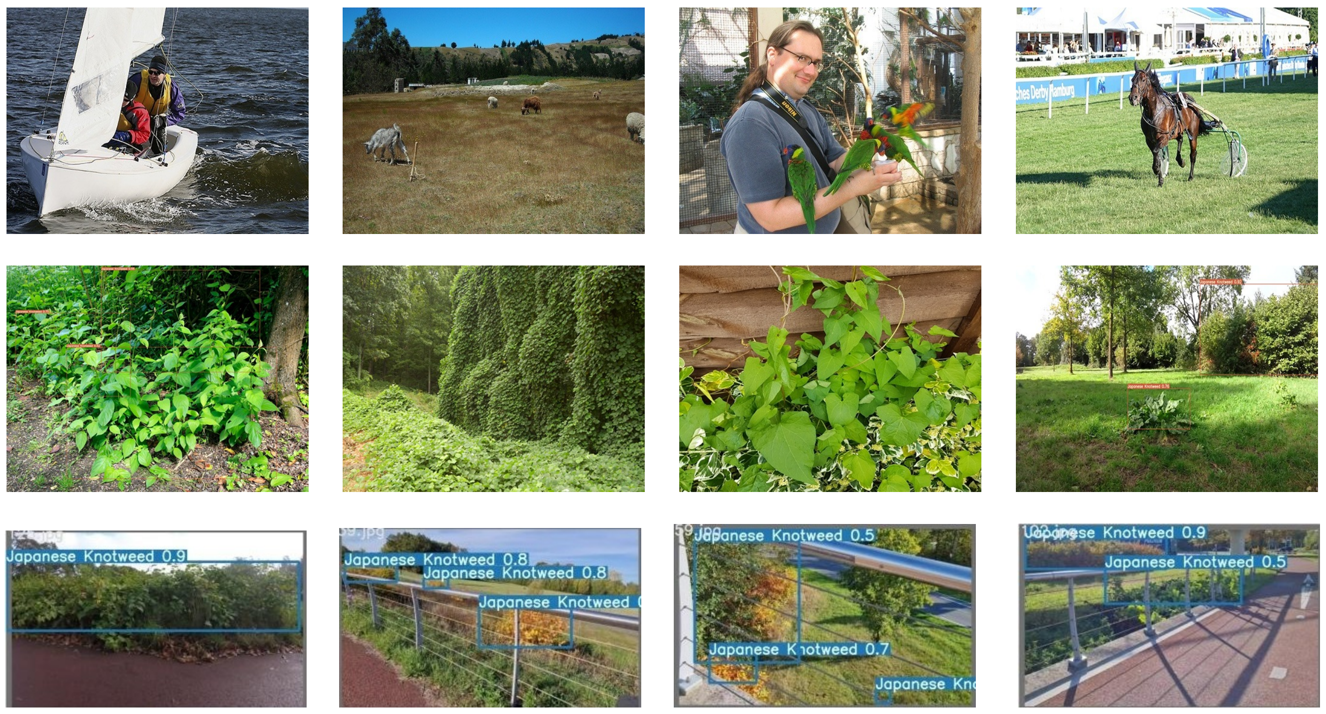


Fig. 16: The figure shows how the model generalizes to other classes, different plant species and its predictions on the test set comprising Japanese Knotweed using YOLOv7 model

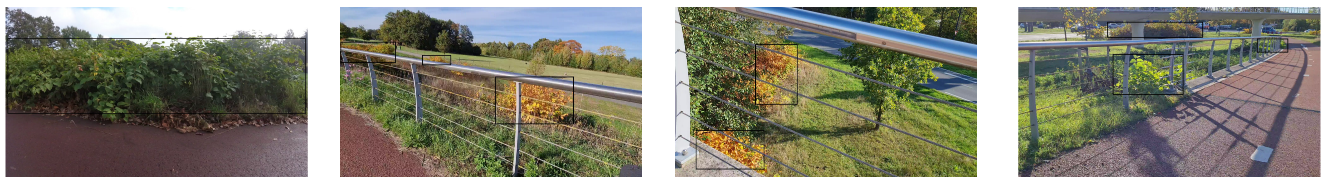


Fig. 17: The figure shows the ground truth for the Japanese Knotweed class for reference comparison with Faster RCNN, DeFRCN, and YOLOv7

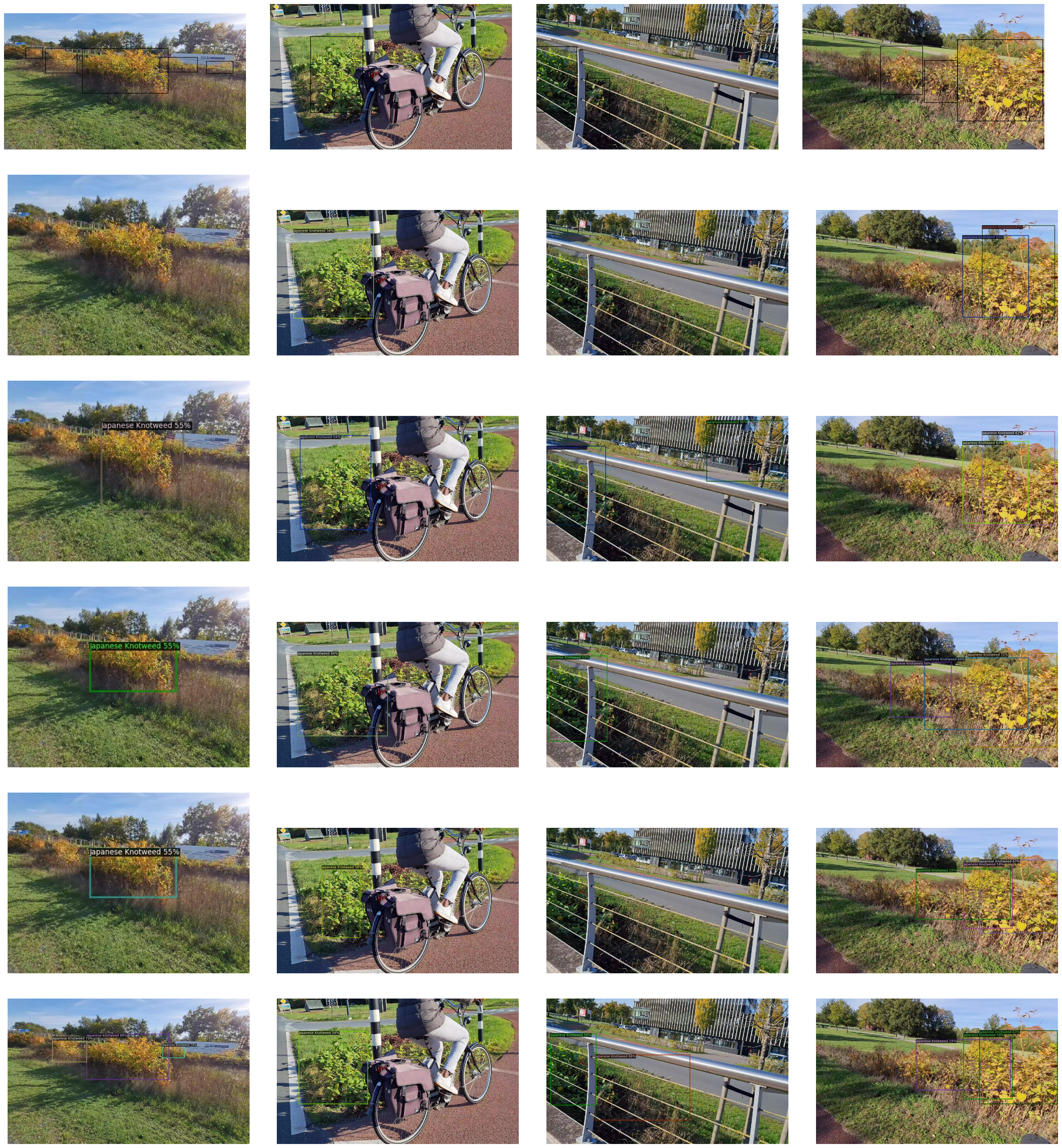


Fig. 18: The figure shows 1, 2, 3, 5, 10 shot model with data augmentation and large support set prediction on the test set comprising the Japanese Knotweed using DeFRCN model on seed 0 as an example. The first row represents ground truth for reference.

References

[1] E. Commission, "Invasive Alien Species." https://ec.europa.eu/environment/nature/invasivealien/index_en.htm. Accessed on April 7, 2023.

[2] W. Universiteit, "Japanese Knotweed: How to Control and Remove It." https://www.wur.nl/en/article/japanese-knotweed-how-to-control-and-remove-it.htm. Accessed on Feb. 5, 2022.

[3] IUCN, "The Global Invasive Species Database: 100 of the World's Worst Invasive Alien Species." http://www.iucngisd.org/gisd/100_worst.php. Accessed on April 7, 2023.

[4] D. Jones, S. Pike, M. Thomas, and D. Murphy, "Object-based image analysis for detection of japanese knotweed s.l. taxa (polygonaceae) in wales (uk)," *Remote Sensing*, vol. 3, no. 2, pp. 319–342, 2011.

[5] A. Smerdu, U. Kanjir, and . Kokalj, "Automatic detection of japanese knotweed in urban areas from aerial and satellite data (special issue: Detection and control of alien forest species in a changing world)," *Management of Biological Invasions*, vol. 11, pp. 661–676, 10 2020.

[6] S. Shanmugam, E. Assuno, R. Mesquita, A. Veiros, and P. Gaspar, "Automated weed detection systems: A review," *Kinet. Energ. Environ. Sustain.*, vol. 5, no. 6, pp. 271–284, 2020.

[7] X. Qiao, Y.-Z. Li, G.-Y. Su, H.-K. Tian, S. Zhang, Z. Sun, L. Yang, F.-H. Wan, and W.-Q. Qian, "Mmnet: Identifying mikania micrantha kunth in the wild via a deep convolutional neural network," *Journal of Integrative Agriculture*, vol. 19, no. 5, pp. 1292–1300, 2020.

[8] M. Dyrmann, A. K. Mortensen, L. Linneberg, T. T. Hye, and K. Bjerge, "Camera assisted roadside monitoring for invasive alien plant species using deep learning," *Sensors*, vol. 21, no. 18, 2021.

[9] W. Qian, Y. Huang, Q. Liu, W. Fan, Z. Sun, H. Dong, F. Wan, and X. Qiao, "Uav and a deep convolutional neural network for monitoring invasive alien plants in the wild," *Computers and Electronics in Agriculture*, vol. 174, p. 105519, 2020.

[10] D. Jones, S. Pike, T. Malcolm, and D. Murphy, "Object-based image analysis for detection of japanese knotweed s.l. taxa (polygonaceae) in wales (uk)," *Remote Sensing*, vol. 3, 02 2011.

[11] D. Lee, D. J. Hess, and M. A. Heldeweg, "Safety and privacy regulations for unmanned aerial vehicles: A multiple comparative analysis," *Technology in Society*, vol. 71, p. 102079, 2022.

[12] M. Gao, C. Hugenholtz, T. Fox, M. Kucharczyk, T. Barchyn, and P. Nesbit, "Weather constraints on global drone flyability," *Scientific Reports*, vol. 11, p. 12092, 06 2021.

[13] J. Ahmad, K. Muhammad, I. Ahmad, W. Ahmad, M. L. Smith, L. N. Smith, D. K. Jain, H. Wang, and I. Mehmood, "Visual features based boosted classification of weeds for real-time selective herbicide sprayer systems," *Computers in Industry*, vol. 98, pp. 23–33, 2018.

[14] K. James and K. Bradshaw, "Detecting plant species in the field with deep learning and drone technology," *Methods in Ecology and Evolution*, vol. 11, 08 2020.

[15] S. Kentsch, M. Cabezas, L. Tomhave, J. Gro, B. Burkhard, M. L. Lopez Caceres, K. Waki, and Y. Diez, "Analysis of uav-acquired wetland orthomosaics using gis, computer vision, computational topology and deep learning," *Sensors*, vol. 21, no. 2, 2021.

[16] B. Ashqar and S. Abu-Naser, "Identifying images of invasive hydrangea using pre-trained deep convolutional neural networks," *International Journal for Academic Development*, vol. 3, 04 2019.

[17] R. O. Q. Dias and D. L. Borges, "Recognizing plant species in the wild: Deep learning results and a new database," in *2016 IEEE International Symposium on Multimedia (ISM)*, pp. 197–202, 2016.

[18] Q. Wang, M. Cheng, S. Huang, Z. Cai, J. Zhang, and H. Yuan, "A deep learning approach incorporating yolo v5 and attention mechanisms for field real-time detection of the invasive weed solanum rostratum dunal seedlings," *Computers and Electronics in Agriculture*, vol. 199, p. 107194, 08 2022.

[19] N. Elias, "Deep learning methodology for early detection and outbreak prediction of invasive species growth," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 6335–6343, January 2023.

[20] K. Ozcan, A. Sharma, S. Bradbury, D. Schweitzer, T. Blader, and S. Blodgett, "Milkweed (asclepias syriaca) plant detection using mobile cameras," *Ecosphere*, vol. 11, 01 2020.

[21] Y. Jiang, J. Zhang, and J. Wang, "Detection of eupatorium adenophorum based on migration learning," in *Proceedings of the 2020 6th International Conference on Computing and Artificial Intelligence*, ICCAI '20, (New York, NY, USA), p. 218222, Association for Computing Machinery, 2020.

[22] B. Kang, Z. Liu, X. Wang, F. Yu, J. Feng, and T. Darrell, "Few-shot object detection via feature reweighting," in *2019 IEEE International Conference on Computer Vision (ICCV)*, pp. 952–961, IEEE, 2019.

[23] Q. Dou, S. Qiao, W. Liu, T. Yao, J. T. Kwok, and C. C. Loy, "Frustratingly simple few-shot object detection," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12625–12634, IEEE, 2020.

[24] Y. Li, L. Wei, J. Xu, Y. Yang, Z. Deng, and W. Liu, "Few-shot object detection via association and discrimination," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12635–12644, IEEE, 2020.

[25] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *CoRR*, vol. abs/1506.01497, 2015.

[26] C.-Y. Wang, A. Bochkovskiy, and H.-y. Liao, "Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," 07 2022.

[27] W. Zhang, Y. Chen, J. Xiao, K. Liu, and M. Yang, "An empirical study and comparison of recent few-shot object detection algorithms," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 132–133, 2020.

[28] Y. Liu, J. Yang, Y. Wang, and J. Zhu, "Few-shot object detection via association and discrimination," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14873–14882, 2021.

[29] L. Qiao, Y. Zhao, Z. Li, X. Qiu, J. Wu, and C. Zhang, "Defrcn: Decoupled faster R-CNN for few-shot object detection," *CoRR*, vol. abs/2108.09017, 2021.

[30] H. Müller, P. Clough, T. Deselaers, B. Kämpgen, A. Doucet, S. Hillion, W. Hersh, W. J. Long, and G. J. F. Jones, "ImageCLEF: experimental evaluation in visual information retrieval," in *Advances in Information Retrieval*, vol. 4956 of *Lecture Notes in Computer Science*, pp. 685–692, Springer Berlin Heidelberg, 2008.

[31] GBIF Secretariat, "Gbif backbone taxonomy." Checklist dataset, 2021.

[32] iNaturalist, "inaturalist research-grade observations." Occurrence dataset, 2021.

[33] S. Branson, J. D. Wegner, D. Hall, N. Lang, K. Schindler, and P. Perona, "From google maps to a fine-grained catalog of street trees," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2014–2022, 2015.

[34] B. Sekachev, N. Manovich, M. Zhiltsov, A. Zhavoronkov, D. Kalinin, B. Hoff, TOsmanov, D. Kruchinin, A. Zankevich, DmitriySidnev, M. Markelov, Johannes222, M. Chenuet, a andre, telenachos, A. Melnikov, J. Kim, L. Ilouz, N. Glazov, Priya4607, R. Tehrani, S. Jeong, V. Skubriev, S. Yonekura, vugia truong, zliang7, lizhming, and T. Truong, "opencv/cvat: v1.1.0," Aug. 2020.

[35] C. Shorten and T. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, 07 2019.

[36] R. Padilla, S. Netto, and E. da Silva, "A survey on performance metrics for object-detection algorithms," 07 2020.

[37] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, "Detectron2," *GitHub repository*, vol. 3, 2019.

[38] M. Dyrmann, A. K. Mortensen, L. Linneberg, T. T. Hye, and K. Bjerge, "Camera assisted roadside monitoring for invasive alien plant species using deep learning," *Sensors*, vol. 21, no. 18, 2021.

[39] M. M. Bejani and M. Ghatee, "A systematic review on overfitting control in shallow and deep neural networks," *Artificial Intelligence Review*, pp. 1–48, 2021.

[40] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?," *CoRR*, vol. abs/1411.1792, 2014.

[41] E. Muyue, "Defrcn." https://github.com/er-muyue/DeFRCN, 2019.

[42] "VOC: tensorflow datasets." Accessed on May 15, 2023.

[43] WongKinYiu, "YOLOv7: Real-time Object Detection." https://github.com/WongKinYiu/yolov7, 2021.

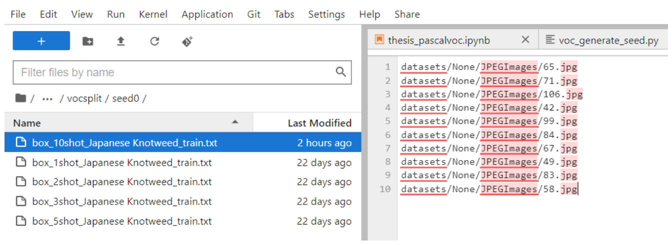
[44] A. Bochkovskiy, C.-Y. Wang, and H.-y. Liao, "Yolov4: Optimal speed and accuracy of object detection," 04 2020.

Appendix

This section provides results, additional information for increasing the understandability of the research paper, and other experiments and few code fragments.

A. Few-shot object detection algorithm: Decoupled Faster RCNN (DeFRCN)

1) Random Seed Generator: Single & Multiple Instance: The DeFRCN repository lacks scripts for organizing the dataset in multiple seeds with experiments 1, 2, 3, 5, 10 and 88 shots. Thus, the script was created. In total, there are 30 seeds. An example of seed 0 with all experiments has been presented in the figure 19.

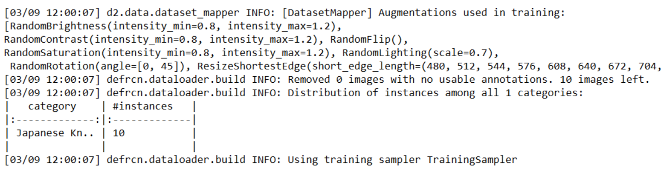


Name	Last Modified
box_10shot_Japanese Knotweed_train.txt	2 hours ago
box_1shot_Japanese Knotweed_train.txt	22 days ago
box_2shot_Japanese Knotweed_train.txt	22 days ago
box_3shot_Japanese Knotweed_train.txt	22 days ago
box_5shot_Japanese Knotweed_train.txt	22 days ago

```
datasets/None/JPEGImages/65.jpg
datasets/None/JPEGImages/71.jpg
datasets/None/JPEGImages/106.jpg
datasets/None/JPEGImages/42.jpg
datasets/None/JPEGImages/99.jpg
datasets/None/JPEGImages/84.jpg
datasets/None/JPEGImages/67.jpg
datasets/None/JPEGImages/49.jpg
datasets/None/JPEGImages/83.jpg
datasets/None/JPEGImages/58.jpg
```

Fig. 19: An example of seed 0 with multiple experiments containing 1, 2, 3, 5, 10 random images

For each experiment, a respective number of instance has been used to train the model, as suggested by the DeFRCN research paper. For instance, figure 23 gives illustrate seed 0 and 10 shot as an example case which uses 10 instances. Furthermore, for 88 shots complete, all the instance of 88 image has been used.



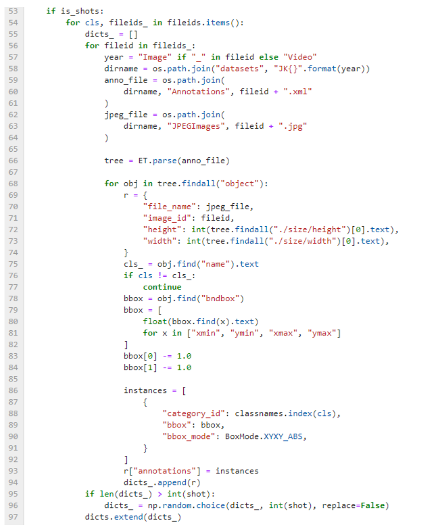
```
[03/09 12:00:07] d2.data.dataset_mapper INFO: [DatasetMapper] Augmentations used in training:
[RandomBrightness(intensity_min=0.8, intensity_max=1.2),
RandomContrast(intensity_min=0.8, intensity_max=1.2), RandomFlip(),
RandomSaturation(intensity_min=0.8, intensity_max=1.2), RandomLighting(scale=0.7),
 RandomRotation(angle=[0, 45]), ResizeShortestEdge(short_edge_length=(480, 512, 544, 576, 608, 640, 672, 704,
[03/09 12:00:07] defrcn.dataloader.build INFO: Removed 0 images with no usable annotations. 10 images left.
[03/09 12:00:07] defrcn.dataloader.build INFO: Distribution of instances among all 1 categories:
|   category    | #instances    |
|:-------------:|:-------------|
| Japanese Kn.. | 10            |
|               |               |
[03/09 12:00:07] defrcn.dataloader.build INFO: Using training sampler TrainingSampler
```

Fig. 20: An example of 10 shot seed0 experiment with 10 instances

2) Dataset Loader: Detectron2 dataset loader is build on top of PyTorch 'DataLoader' class that returns a list of images in the dataset and their corresponding annotation. For creating a dataset loader, two approaches were explored and implemented:

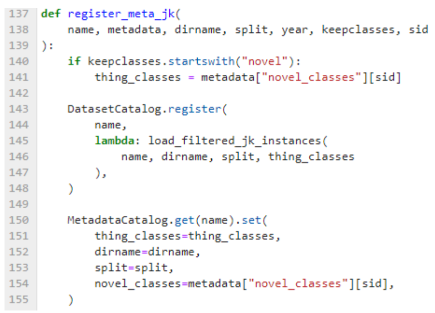
1) Creating a custom data loader for the Japanese Knotweed
2) Using the existing implementation by extending it to include Japanese Knotweed

One advantage of extending the implementation is the ability to use the base class for generalized few-shot object detection (GFSOD) on both the base and novel classes for multi-class object detection. This makes organizing the dataset hassle-free, as we need to follow the same directory hierarchy as mentioned by the DeFRCN. This method was used in this research. The custom data loader, was still implemented to understand the adaptability of the existing repository.



```python
    if is_shots:
        for cls, fileids_ in fileids.items():
            dicts_ = []
            for fileid in fileids_:
                year = "Image" if "_" in fileid else "Video"
                dirname = os.path.join("datasets", "JK{}".format(year))
                anno_file = os.path.join(
                    dirname, "Annotations", fileid + ".xml"
                )
                jpeg_file = os.path.join(
                    dirname, "JPEGImages", fileid + ".jpg"
                )

                tree = ET.parse(anno_file)

                for obj in tree.findall("object"):
                    r = {
                        "file_name": jpeg_file,
                        "image_id": fileid,
                        "height": int(tree.findall("./size/height")[0].text),
                        "width": int(tree.findall("./size/width")[0].text),
                    }
                    cls_ = obj.find("name").text
                    if cls != cls_:
                        continue
                    bbox = obj.find("bndbox")
                    bbox = [
                        float(bbox.find(x).text)
                        for x in ["xmin", "ymin", "xmax", "ymax"]
                    ]
                    bbox[0] -= 1.0
                    bbox[1] -= 1.0

                    instances = [
                        {
                            "category_id": classnames.index(cls),
                            "bbox": bbox,
                            "bbox_mode": BoxMode.XYXY_ABS,
                        }
                    ]
                    r["annotations"] = instances
                    dicts_.append(r)
            if len(dicts_) > int(shot):
                dicts_ = np.random.choice(dicts_, int(shot), replace=False)
            dicts.extend(dicts_)
```

Fig. 21: A custom dataset loader for Japanese Knotweed



```python
def register_meta_jk(
    name, metadata, dirname, split, year, keepclasses, sid
):
    if keepclasses.startswith("novel"):
        thing_classes = metadata["novel_classes"][sid]

    DatasetCatalog.register(
        name,
        lambda: load_filtered_jk_instances(
            name, dirname, split, thing_classes
        ),
    )

    MetadataCatalog.get(name).set(
        thing_classes=thing_classes,
        dirname=dirname,
        split=split,
        novel_classes=metadata["novel_classes"][sid],
    )
```

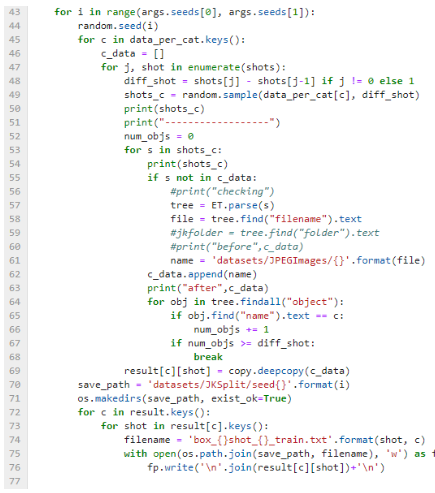
Fig. 22: Registering Metadata and Dataset under Detectron2

3) Validation Hook: DeFRCN directory doesn't have validation logging. Thus, a validation hook script was created to log validation loss. The figure 48 describe the logging for the validation loss for 10 shot seed 0 as an example. The figure 49 shows the validation loss curve of 10 shot with data augmentation (pink) and without data augmentation (purple).

4) A custom datamapper: A custom datamapper for online data augmentation with manual seed* (not to be confused with experimental seed). This manual seed* ensures the same augmentation is applied across multiple experiments and across seeds. Figure 24 shows the code snippet for the same.

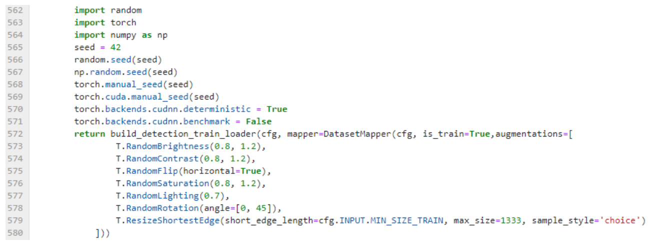
5) Visualization script: The DeFRCN repository doesn't have a script for visualization of the prediction. It enables to examine the model performance, helps in identifying and diagnosing specific issues with the model. It is possible, for instance, to intuitively identify cases in which the model makes false positives or false negative. This helps in refining and improving the model performance. Thus, a visualization script is made for training and prediction.

6) A configuration file for custom 88 shots: A configuration file is created based on the implementation of the 10 shot. It



```python
    for i in range(args.seeds[0], args.seeds[1]):
        random.seed(i)
        for c in data_per_cat.keys():
            c_data = []
            for j, shot in enumerate(shots):
                diff_shot = shots[j] - shots[j-1] if j != 0 else 1
                shots_c = random.sample(data_per_cat[c], diff_shot)
                print(shots_c)
                print("------------------")
                num_objs = 0
                for s in shots_c:
                    print(shots_c)
                    if s not in c_data:
                        #print("checking")
                        tree = ET.parse(s)
                        file = tree.find("filename").text
                        #jkfolder = tree.find("folder").text
                        #print("before",c_data)
                        name = 'datasets/JPEGImages/{}'.format(file)
                    c_data.append(name)
                    print("after",c_data)
                    for obj in tree.findall("object"):
                        if obj.find("name").text == c:
                            num_objs += 1
                        if num_objs >= diff_shot:
                            break
                result[c][shot] = copy.deepcopy(c_data)
        save_path = 'datasets/JKSplit/seed{}'.format(i)
        os.makedirs(save_path, exist_ok=True)
        for c in result.keys():
            for shot in result[c].keys():
                filename = 'box_{}shot_{}_train.txt'.format(shot, c)
                with open(os.path.join(save_path, filename), 'w') as fp:
                    fp.write('\n'.join(result[c][shot])+'\n')
```

Fig. 23: Creating multiple dataset seeds for 1, 2, 3, 5, 10 experiments

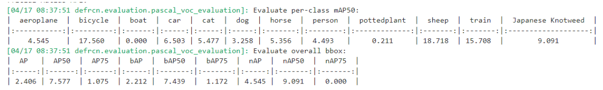


```python
        import random
        import torch
        import numpy as np
        seed = 42
        random.seed(seed)
        np.random.seed(seed)
        torch.manual_seed(seed)
        torch.cuda.manual_seed(seed)
        torch.backends.cudnn.deterministic = True
        torch.backends.cudnn.benchmark = False
        return build_detection_train_loader(cfg, mapper=DatasetMapper(cfg, is_train=True,augmentations=[
                T.RandomBrightness(0.8, 1.2),
                T.RandomContrast(0.8, 1.2),
                T.RandomFlip(horizontal=True),
                T.RandomSaturation(0.8, 1.2),
                T.RandomLighting(0.7),
                T.RandomRotation(angle=[0, 45]),
                T.ResizeShortestEdge(short_edge_length=cfg.INPUT.MIN_SIZE_TRAIN, max_size=1333, sample_style='choice')
            ]))
```

Fig. 24: Custom datamapper with manual seed and online data augmentation

is essentially a YAML file with different hyperparameter such as learning rate, iteration, decoupling coefficient for RPN and RCNN, freezing layers etc.

7) Multi-Class: For performing a multi-class object detection, a GFSOD approach was used. GFSOD approach gives performance on both novel class and base class. An example of multi-class using 10 shot on seed 0 with data augmentation is shown in the below 25.



```
[04/17 08:37:51 defrcn.evaluation.pascal_voc_evaluation]: Evaluate per-class mAP50:
|  aeroplane  |  bicycle  |  boat  |  car  |  cat  |  dog  |  horse  |  person  |  pottedplant  |  sheep  |  train  |  Japanese Knotweed  |
|:-----------:|:---------:|:------:|:-----:|:-----:|:-----:|:-------:|:--------:|:-------------:|:-------:|:-------:|:-------------------:|
|    4.545    |  17.560   | 0.000  | 6.503 | 5.477 | 3.258 |  5.356  |  4.493   |     0.211     | 18.718  | 15.708  |        9.091        |
[04/17 08:37:51 defrcn.evaluation.pascal_voc_evaluation]: Evaluate overall bbox:
|  AP   |  AP50  |  AP75  |  bAP  |  bAP50  |  bAP75  |  nAP  |  nAP50  |  nAP75  |
|:-----:|:------:|:------:|:-----:|:-------:|:-------:|:-----:|:-------:|:-------:|
| 2.406 | 7.577  | 1.075  | 2.212 |  7.439  |  1.172  | 4.545 |  9.091  |  0.000  |
```

Fig. 25: An example of multi-class 10 shot on seed 0 and data augmentation on test set

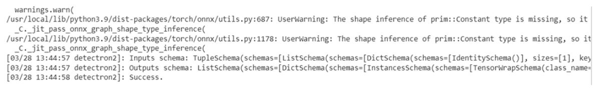
8) Overfitting: The DeFRCN avoids overfitting on a small dataset. This can be verified from the figure 53, 52, 51, 50, 49, 54. Under the same set of learning rate 1xe^{-4}, the Faster RCNN model still overfits.

B. Deployment

For deployment, detectron2 offers three methods which are tracing, scripting and caffe_tracing. Caffe_tracing supports ONNX format. It allows interoperability between different machine learning frameworks and deployment in hardware platform. If the hardware platform doesn't have ONNX support, necessary conversion needs to be performed. For instance, if the hardware choice is TinyML by Arduino, the ONNX formatted model needs to be converted to TensorFlow lite. Thus, to deploy the model from Detectron2 to TensorFlow lite. This pipeline needs to be follow: Detectron2 $\rightarrow$ ONNX $\rightarrow$ TensorFlow$\rightarrow$ TensorFlow lite

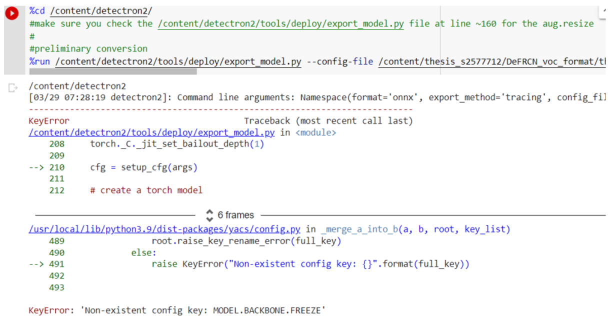
However, detectron2 has depreciated the support for caffe2_tracing. Thus, the other two alternatives needs to be followed. The pipeline for which is: Detectron2 $\rightarrow$ PyTorch $\rightarrow$ ONNX $\rightarrow$ TensorFlow$\rightarrow$ TensorFlow lite

However, these options are only available for the standard model of the detectron2 model zoo as shown in the figure 26. DeFRCN which has made architectural changes to the standard model with GDL and PCB are not recognizable as shown in 27. Thus, it requires development of the novel deployment script dedicated to DeFRCN.



```
  warnings.warn(
/usr/local/lib/python3.9/dist-packages/torch/onnx/utils.py:687: UserWarning: The shape inference of prim::Constant type is missing, so it
  _C._jit_pass_onnx_graph_shape_type_inference(
/usr/local/lib/python3.9/dist-packages/torch/onnx/utils.py:1178: UserWarning: The shape inference of prim::Constant type is missing, so it
  _C._jit_pass_onnx_graph_shape_type_inference(
[03/28 13:44:57 detectron2]: Inputs schema: TupleSchema(schemas=[ListSchema(schemas=[DictSchema(schemas=[IdentitySchema()], sizes=[1], key
[03/28 13:44:57 detectron2]: Outputs schema: ListSchema(schemas=[DictSchema(schemas=[InstancesSchema(schemas=[TensorWrapSchema(class_name=
[03/28 13:44:58 detectron2]: Success.
```

Fig. 26: Successful conversion of model deployment using Faster RCNN model



```
%cd /content/detectron2/
#make sure you check the /content/detectron2/tools/deploy/export_model.py file at line ~160 for the aug.resize
#
#preliminary conversion
%run /content/detectron2/tools/deploy/export_model.py --config-file /content/thesis_s2577712/DeFRCN_voc_format/tl

/content/detectron2
[03/29 07:28:19 detectron2]: Command line arguments: Namespace(format='onnx', export_method='tracing', config_fil
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
/content/detectron2/tools/deploy/export_model.py in <module>
    208     torch._C._jit_set_bailout_depth(1)
    209 
--> 210     cfg = setup_cfg(args)
    211 
    212     # create a torch model

6 frames
/usr/local/lib/python3.9/dist-packages/yacs/config.py in _merge_a_into_b(a, b, root, key_list)
    489                 root.raise_key_rename_error(full_key)
    490             else:
--> 491                 raise KeyError("Non-existent config key: {}".format(full_key))
    492 
    493 

KeyError: 'Non-existent config key: MODEL.BACKBONE.FREEZE'
```

Fig. 27: Failed conversion of model deployment using DeFRCN model

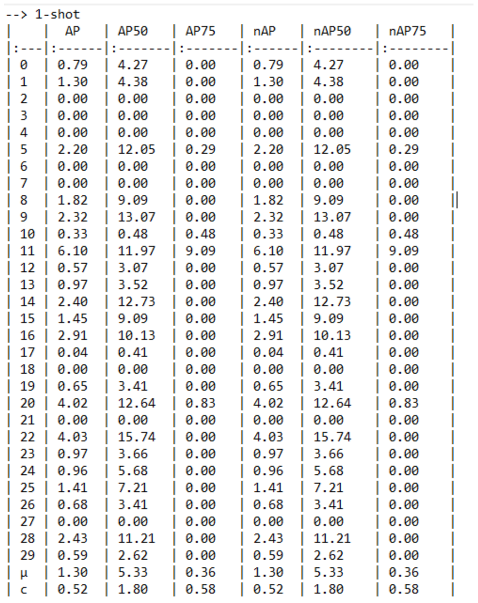
1) Results: For results concerning multiple seed evaluation, only mean (μ) results over 30 seeds are shown. An example of how the 30 seed evaluation results generated for 1 shot is shown in the figure 28.

C. Faster RCNN

For reducing the false positive, background images (11 images consisting of trees and grass which indicate the natural environment of Japanese Knotweed beside the road) were additionally added with 88 shot experiment. However, no performance gain was observed. The code snippet is shown in the figure 47.

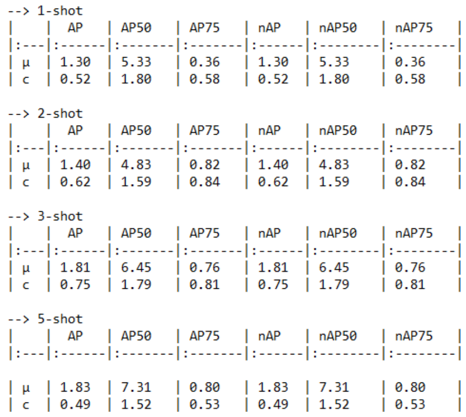
D. YOLOv7

For deciding the layers to freeze, three experiments were carried out: 1) Freezing all the layers (50) of the network, 2) Freezing 50% of layers of the network, and 3) No freezing of the network. The corresponding mAP$_{50}$ on the validation set was observed. Experiments 1 and 3 reduced the mAP$_{50}$, therefore, Exp2 was selected to conduct the research.



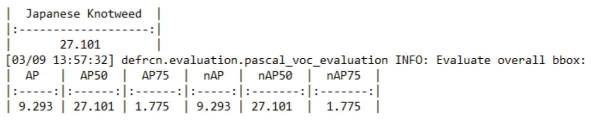
```
--> 1-shot
|    | AP    | AP50   | AP75   | nAP   | nAP50   | nAP75   |
|:---|:------|:-------|:-------|:------|:--------|:--------|
| 0  | 0.79  | 4.27   | 0.00   | 0.79  | 4.27    | 0.00    |
| 1  | 1.30  | 4.38   | 0.00   | 1.30  | 4.38    | 0.00    |
| 2  | 0.00  | 0.00   | 0.00   | 0.00  | 0.00    | 0.00    |
| 3  | 0.00  | 0.00   | 0.00   | 0.00  | 0.00    | 0.00    |
| 4  | 0.00  | 0.00   | 0.00   | 0.00  | 0.00    | 0.00    |
| 5  | 2.20  | 12.05  | 0.29   | 2.20  | 12.05   | 0.29    |
| 6  | 0.00  | 0.00   | 0.00   | 0.00  | 0.00    | 0.00    |
| 7  | 0.00  | 0.00   | 0.00   | 0.00  | 0.00    | 0.00    |
| 8  | 1.82  | 9.09   | 0.00   | 1.82  | 9.09    | 0.00    |
| 9  | 2.32  | 13.07  | 0.00   | 2.32  | 13.07   | 0.00    |
| 10 | 0.33  | 0.48   | 0.48   | 0.33  | 0.48    | 0.48    |
| 11 | 6.10  | 11.97  | 9.09   | 6.10  | 11.97   | 9.09    |
| 12 | 0.57  | 3.07   | 0.00   | 0.57  | 3.07    | 0.00    |
| 13 | 0.97  | 3.52   | 0.00   | 0.97  | 3.52    | 0.00    |
| 14 | 2.40  | 12.73  | 0.00   | 2.40  | 12.73   | 0.00    |
| 15 | 1.45  | 9.09   | 0.00   | 1.45  | 9.09    | 0.00    |
| 16 | 2.91  | 10.13  | 0.00   | 2.91  | 10.13   | 0.00    |
| 17 | 0.04  | 0.41   | 0.00   | 0.04  | 0.41    | 0.00    |
| 18 | 0.00  | 0.00   | 0.00   | 0.00  | 0.00    | 0.00    |
| 19 | 0.65  | 3.41   | 0.00   | 0.65  | 3.41    | 0.00    |
| 20 | 4.02  | 12.64  | 0.83   | 4.02  | 12.64   | 0.83    |
| 21 | 0.00  | 0.00   | 0.00   | 0.00  | 0.00    | 0.00    |
| 22 | 4.03  | 15.74  | 0.00   | 4.03  | 15.74   | 0.00    |
| 23 | 0.97  | 3.66   | 0.00   | 0.97  | 3.66    | 0.00    |
| 24 | 0.96  | 5.68   | 0.00   | 0.96  | 5.68    | 0.00    |
| 25 | 1.41  | 7.21   | 0.00   | 1.41  | 7.21    | 0.00    |
| 26 | 0.68  | 3.41   | 0.00   | 0.68  | 3.41    | 0.00    |
| 27 | 0.00  | 0.00   | 0.00   | 0.00  | 0.00    | 0.00    |
| 28 | 2.43  | 11.21  | 0.00   | 2.43  | 11.21   | 0.00    |
| 29 | 0.59  | 2.62   | 0.00   | 0.59  | 2.62    | 0.00    |
| μ  | 1.30  | 5.33   | 0.36   | 1.30  | 5.33    | 0.36    |
| c  | 0.52  | 1.80   | 0.58   | 0.52  | 1.80    | 0.58    |
```

Fig. 28: An example of 1 shot with 30 seeds and data augmentation on test set



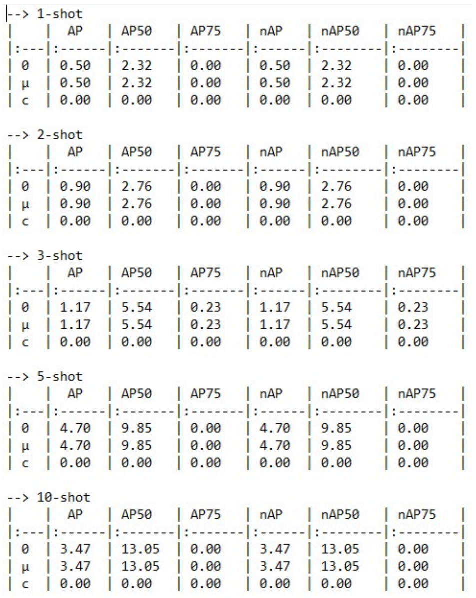
```
--> 1-shot
|    | AP    | AP50   | AP75   | nAP   | nAP50   | nAP75   |
|:---|:------|:-------|:-------|:------|:--------|:--------|
| μ  | 1.30  | 5.33   | 0.36   | 1.30  | 5.33    | 0.36    |
| c  | 0.52  | 1.80   | 0.58   | 0.52  | 1.80    | 0.58    |

--> 2-shot
|    | AP    | AP50   | AP75   | nAP   | nAP50   | nAP75   |
|:---|:------|:-------|:-------|:------|:--------|:--------|
| μ  | 1.40  | 4.83   | 0.82   | 1.40  | 4.83    | 0.82    |
| c  | 0.62  | 1.59   | 0.84   | 0.62  | 1.59    | 0.84    |

--> 3-shot
|    | AP    | AP50   | AP75   | nAP   | nAP50   | nAP75   |
|:---|:------|:-------|:-------|:------|:--------|:--------|
| μ  | 1.81  | 6.45   | 0.76   | 1.81  | 6.45    | 0.76    |
| c  | 0.75  | 1.79   | 0.81   | 0.75  | 1.79    | 0.81    |

--> 5-shot
|    | AP    | AP50   | AP75   | nAP   | nAP50   | nAP75   |
|:---|:------|:-------|:-------|:------|:--------|:--------|

| μ  | 1.83  | 7.31   | 0.80   | 1.83  | 7.31    | 0.80    |
| c  | 0.49  | 1.52   | 0.53   | 0.49  | 1.52    | 0.53    |
```

Fig. 29: An aggregated example of 1, 2, 3, 5 on 30 seeds with data augmentation



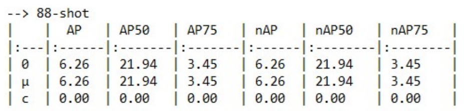
```
|  Japanese Knotweed  |
|:-------------------:|
|       27.101        |
[03/09 13:57:32] defrcn.evaluation.pascal_voc_evaluation INFO: Evaluate overall bbox:
|  AP   |  AP50   |  AP75   |  nAP  |  nAP50   |  nAP75   |
|:-----:|:-------:|:-------:|:-----:|:--------:|:--------:|
| 9.293 | 27.101 | 1.775   | 9.293 | 27.101   |  1.775   |
```

Fig. 30: An aggregated example of 1, 2, 3, 5 on 30 seeds without data augmentation



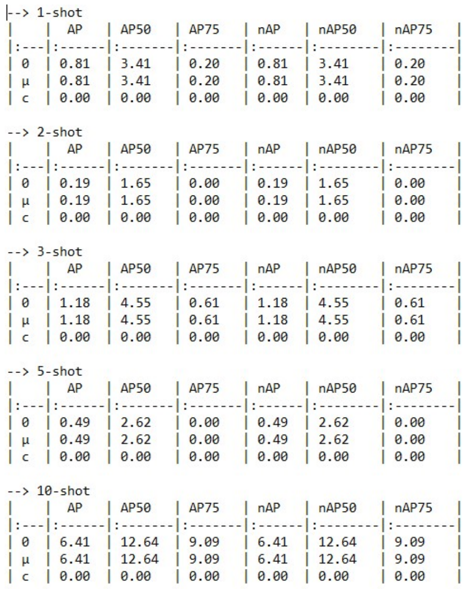
```
--> 1-shot
|    | AP    | AP50   | AP75   | nAP   | nAP50   | nAP75   |
|:---|:------|:-------|:-------|:------|:--------|:--------|
| 0  | 0.50  | 2.32   | 0.00   | 0.50  | 2.32    | 0.00    |
| μ  | 0.50  | 2.32   | 0.00   | 0.50  | 2.32    | 0.00    |
| c  | 0.00  | 0.00   | 0.00   | 0.00  | 0.00    | 0.00    |

--> 2-shot
|    | AP    | AP50   | AP75   | nAP   | nAP50   | nAP75   |
|:---|:------|:-------|:-------|:------|:--------|:--------|
| 0  | 0.90  | 2.76   | 0.00   | 0.90  | 2.76    | 0.00    |
| μ  | 0.90  | 2.76   | 0.00   | 0.90  | 2.76    | 0.00    |
| c  | 0.00  | 0.00   | 0.00   | 0.00  | 0.00    | 0.00    |

--> 3-shot
|    | AP    | AP50   | AP75   | nAP   | nAP50   | nAP75   |
|:---|:------|:-------|:-------|:------|:--------|:--------|
| 0  | 1.17  | 5.54   | 0.23   | 1.17  | 5.54    | 0.23    |
| μ  | 1.17  | 5.54   | 0.23   | 1.17  | 5.54    | 0.23    |
| c  | 0.00  | 0.00   | 0.00   | 0.00  | 0.00    | 0.00    |

--> 5-shot
|    | AP    | AP50   | AP75   | nAP   | nAP50   | nAP75   |
|:---|:------|:-------|:-------|:------|:--------|:--------|
| 0  | 4.70  | 9.85   | 0.00   | 4.70  | 9.85    | 0.00    |
| μ  | 4.70  | 9.85   | 0.00   | 4.70  | 9.85    | 0.00    |
| c  | 0.00  | 0.00   | 0.00   | 0.00  | 0.00    | 0.00    |

--> 10-shot
|    | AP    | AP50   | AP75   | nAP   | nAP50   | nAP75   |
|:---|:------|:-------|:-------|:------|:--------|:--------|
| 0  | 3.47  | 13.05  | 0.00   | 3.47  | 13.05   | 0.00    |
| μ  | 3.47  | 13.05  | 0.00   | 3.47  | 13.05   | 0.00    |
| c  | 0.00  | 0.00   | 0.00   | 0.00  | 0.00    | 0.00    |
```

Fig. 31: Seed 0 with data augmentation



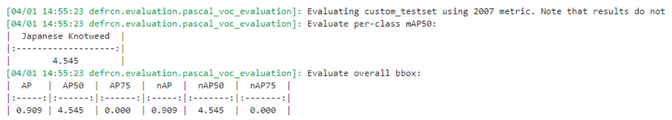
```
--> 88-shot
|    | AP    | AP50   | AP75   | nAP   | nAP50   | nAP75   |
|:---|:------|:-------|:-------|:------|:--------|:--------|
| 0  | 6.26  | 21.94  | 3.45   | 6.26  | 21.94   | 3.45    |
| μ  | 6.26  | 21.94  | 3.45   | 6.26  | 21.94   | 3.45    |
| c  | 0.00  | 0.00   | 0.00   | 0.00  | 0.00    | 0.00    |
```

Fig. 32: 88 shot without data augmentation



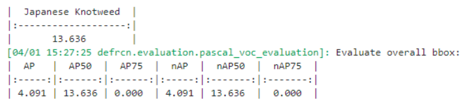
```
--> 1-shot
|    | AP    | AP50   | AP75   | nAP   | nAP50   | nAP75   |
|:---|:------|:-------|:-------|:------|:--------|:--------|
| 0  | 0.81  | 3.41   | 0.20   | 0.81  | 3.41    | 0.20    |
| μ  | 0.81  | 3.41   | 0.20   | 0.81  | 3.41    | 0.20    |
| c  | 0.00  | 0.00   | 0.00   | 0.00  | 0.00    | 0.00    |

--> 2-shot
|    | AP    | AP50   | AP75   | nAP   | nAP50   | nAP75   |
|:---|:------|:-------|:-------|:------|:--------|:--------|
| 0  | 0.19  | 1.65   | 0.00   | 0.19  | 1.65    | 0.00    |
| μ  | 0.19  | 1.65   | 0.00   | 0.19  | 1.65    | 0.00    |
| c  | 0.00  | 0.00   | 0.00   | 0.00  | 0.00    | 0.00    |

--> 3-shot
|    | AP    | AP50   | AP75   | nAP   | nAP50   | nAP75   |
|:---|:------|:-------|:-------|:------|:--------|:--------|
| 0  | 1.18  | 4.55   | 0.61   | 1.18  | 4.55    | 0.61    |
| μ  | 1.18  | 4.55   | 0.61   | 1.18  | 4.55    | 0.61    |
| c  | 0.00  | 0.00   | 0.00   | 0.00  | 0.00    | 0.00    |

--> 5-shot
|    | AP    | AP50   | AP75   | nAP   | nAP50   | nAP75   |
|:---|:------|:-------|:-------|:------|:--------|:--------|
| 0  | 0.49  | 2.62   | 0.00   | 0.49  | 2.62    | 0.00    |
| μ  | 0.49  | 2.62   | 0.00   | 0.49  | 2.62    | 0.00    |
| c  | 0.00  | 0.00   | 0.00   | 0.00  | 0.00    | 0.00    |

--> 10-shot
|    | AP    | AP50   | AP75   | nAP   | nAP50   | nAP75   |
|:---|:------|:-------|:-------|:------|:--------|:--------|
| 0  | 6.41  | 12.64  | 9.09   | 6.41  | 12.64   | 9.09    |
| μ  | 6.41  | 12.64  | 9.09   | 6.41  | 12.64   | 9.09    |
| c  | 0.00  | 0.00   | 0.00   | 0.00  | 0.00    | 0.00    |
```

Fig. 33: Seed 0 without data augmentation



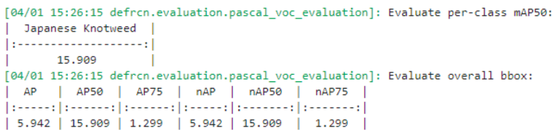
```
[04/01 14:55:23 defrcn.evaluation.pascal_voc_evaluation]: Evaluating custom_testset using 2007 metric. Note that results do not
[04/01 14:55:23 defrcn.evaluation.pascal_voc_evaluation]: Evaluate per-class mAP50:
|  Japanese Knotweed  |
|:-------------------:|
|        4.545        |
[04/01 14:55:23 defrcn.evaluation.pascal_voc_evaluation]: Evaluate overall bbox:
|  AP   |  AP50   |  AP75   |  nAP  |  nAP50   |  nAP75   |
|:-----:|:-------:|:-------:|:-----:|:--------:|:--------:|
| 0.909 | 4.545   | 0.000   | 0.909 |  4.545   |  0.000   |
```

Fig. 34: An example of 1 shot on seed 0 with large support set



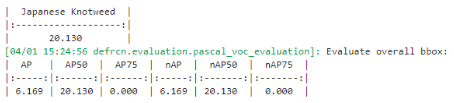
```
|  Japanese Knotweed  |
|:-------------------:|
|       13.636        |
[04/01 15:27:25 defrcn.evaluation.pascal_voc_evaluation]: Evaluate overall bbox:
|  AP   |  AP50  |  AP75  |  nAP  |  nAP50  |  nAP75  |
|:-----:|:------:|:------:|:-----:|:-------:|:-------:|
| 4.091 | 13.636 | 0.000  | 4.091 | 13.636  |  0.000  |
```

Fig. 35: An example of 2 shot on seed 0 with large support set



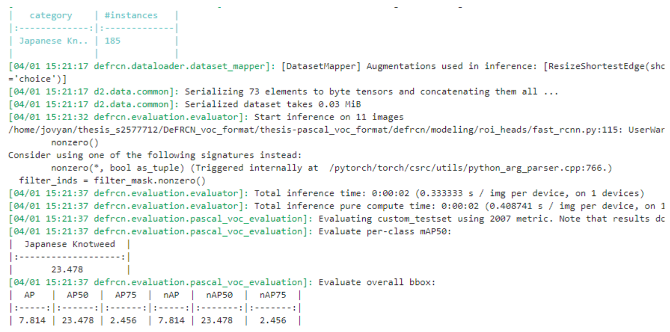
```
[04/01 15:26:15 defrcn.evaluation.pascal_voc_evaluation]: Evaluate per-class mAP50:
|  Japanese Knotweed  |
|:-------------------:|
|       15.909        |
[04/01 15:26:15 defrcn.evaluation.pascal_voc_evaluation]: Evaluate overall bbox:
|  AP   |  AP50  |  AP75  |  nAP  |  nAP50  |  nAP75  |
|:-----:|:------:|:------:|:-----:|:-------:|:-------:|
| 5.942 | 15.909 | 1.299  | 5.942 | 15.909  |  1.299  |
```

Fig. 36: An example of 3 shot on seed 0 with large support set



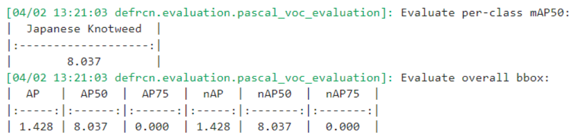
```
|  Japanese Knotweed  |
|:-------------------:|
|       20.130        |
[04/01 15:24:56 defrcn.evaluation.pascal_voc_evaluation]: Evaluate overall bbox:
|  AP   |  AP50  |  AP75  |  nAP  |  nAP50  |  nAP75  |
|:-----:|:------:|:------:|:-----:|:-------:|:-------:|
| 6.169 | 20.130 | 0.000  | 6.169 | 20.130  |  0.000  |
```

Fig. 37: An example of 5 shot on seed 0 with large support set



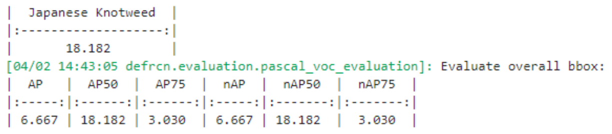
```
|   category    | #instances   |
|:-------------:|:-------------|
| Japanese Kn.. | 185          |
|               |              |
[04/01 15:21:17 defrcn.dataloader.dataset_mapper]: [DatasetMapper] Augmentations used in inference: [ResizeShortestEdge(shc
='choice')]
[04/01 15:21:17 d2.data.common]: Serializing 73 elements to byte tensors and concatenating them all ...
[04/01 15:21:17 d2.data.common]: Serialized dataset takes 0.03 MiB
[04/01 15:21:32 defrcn.evaluation.evaluator]: Start inference on 11 images
/home/jovyan/thesis_s2577712/DeFRCN_voc_format/thesis-pascal_voc_format/defrcn/modeling/roi_heads/fast_rcnn.py:115: UserWar
        nonzero()
Consider using one of the following signatures instead:
        nonzero(*, bool as_tuple) (Triggered internally at  /pytorch/torch/csrc/utils/python_arg_parser.cpp:766.)
  filter_inds = filter_mask.nonzero()
[04/01 15:21:37 defrcn.evaluation.evaluator]: Total inference time: 0:00:02 (0.333333 s / img per device, on 1 devices)
[04/01 15:21:37 defrcn.evaluation.evaluator]: Total inference pure compute time: 0:00:02 (0.408741 s / img per device, on 1
[04/01 15:21:37 defrcn.evaluation.pascal_voc_evaluation]: Evaluating custom_testset using 2007 metric. Note that results dc
[04/01 15:21:37 defrcn.evaluation.pascal_voc_evaluation]: Evaluate per-class mAP50:
|  Japanese Knotweed  |
|:-------------------:|
|       23.478        |
[04/01 15:21:37 defrcn.evaluation.pascal_voc_evaluation]: Evaluate overall bbox:
|  AP   |  AP50  |  AP75  |  nAP  |  nAP50  |  nAP75  |
|:-----:|:------:|:------:|:-----:|:-------:|:-------:|
| 7.814 | 23.478 | 2.456  | 7.814 | 23.478  |  2.456  |
```

Fig. 38: An example of 10 shot on seed 0 with large support set



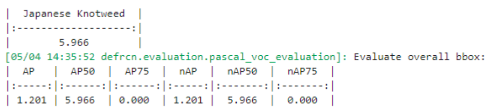
```
[04/02 13:21:03 defrcn.evaluation.pascal_voc_evaluation]: Evaluate per-class mAP50:
|  Japanese Knotweed  |
|:-------------------:|
|       8.037         |
[04/02 13:21:03 defrcn.evaluation.pascal_voc_evaluation]: Evaluate overall bbox:
|  AP   |  AP50  |  AP75  |  nAP  |  nAP50  |  nAP75  |
|:-----:|:------:|:------:|:-----:|:-------:|:-------:|
| 1.428 | 8.037  | 0.000  | 1.428 |  8.037  |  0.000  |
```

Fig. 39: An example of 10 on seed 0 with data augmentation and only GDL



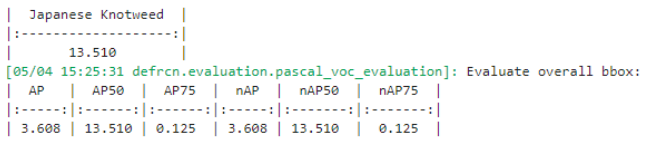
```
|  Japanese Knotweed  |
|:-------------------:|
|       18.182        |
[04/02 14:43:05 defrcn.evaluation.pascal_voc_evaluation]: Evaluate overall bbox:
|  AP   |  AP50  |  AP75  |  nAP  |  nAP50  |  nAP75  |
|:-----:|:------:|:------:|:-----:|:-------:|:-------:|
| 6.667 | 18.182 | 3.030  | 6.667 | 18.182  |  3.030  |
```

Fig. 40: An example of 10 on seed 0 with data augmentation and only PCB



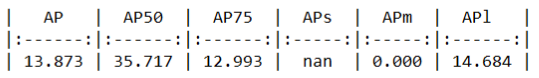
```
|  Japanese Knotweed  |
|:-------------------:|
|       5.966         |
[05/04 14:35:52 defrcn.evaluation.pascal_voc_evaluation]: Evaluate overall bbox:
|  AP   |  AP50  |  AP75  |  nAP  |  nAP50  |  nAP75  |
|:-----:|:------:|:------:|:-----:|:-------:|:-------:|
| 1.201 | 5.966  | 0.000  | 1.201 |  5.966  |  0.000  |
```

Fig. 41: An example of 10 on seed 0 without data augmentation and only GD



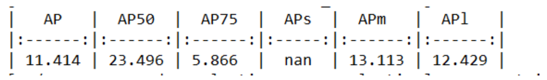
```
|  Japanese Knotweed  |
|:-------------------:|
|       13.510        |
[05/04 15:25:31 defrcn.evaluation.pascal_voc_evaluation]: Evaluate overall bbox:
|  AP   |  AP50  |  AP75  |  nAP  |  nAP50  |  nAP75  |
|:-----:|:------:|:------:|:-----:|:-------:|:-------:|
| 3.608 | 13.510 | 0.125  | 3.608 | 13.510  |  0.125  |
```

Fig. 42: An example of 10 on seed 0 without data augmentation and only PCB



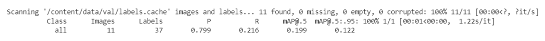
```
|   AP   |  AP50  |  AP75  |  APs  |  APm  |  APl   |
|:------:|:------:|:------:|:-----:|:-----:|:------:|
| 13.873 | 35.717 | 12.993 |  nan  | 0.000 | 14.684 |
```

Fig. 43: Faster RCNN mAP$_{50}$ on 88 shot with data augmentation



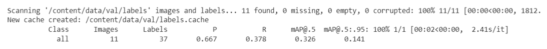
```
|   AP   |  AP50  |  AP75  |  APs  |  APm   |  APl   |
|:------:|:------:|:------:|:-----:|:------:|:------:|
| 11.414 | 23.496 | 5.866  |  nan  | 13.113 | 12.429 |
```

Fig. 44: Faster RCNN mAP$_{50}$ on 88 shot without data augmentation



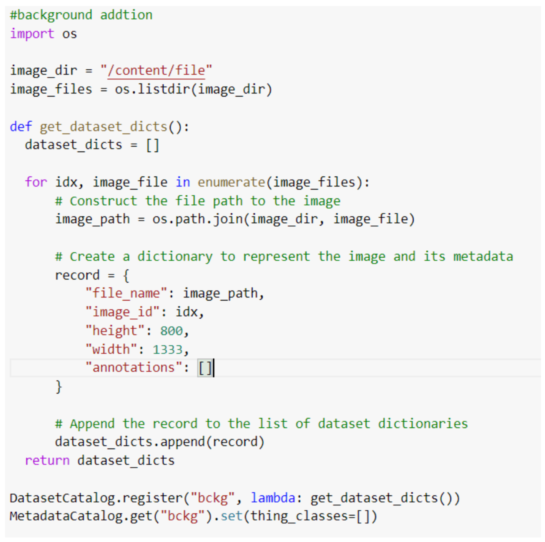
```
Scanning '/content/data/val/labels.cache' images and labels... 11 found, 0 missing, 0 empty, 0 corrupted: 100% 11/11 [00:00<?, ?it/s]
               Class      Images      Labels           P           R      mAP@.5  mAP@.5:.95: 100% 1/1 [00:01<00:00,  1.22s/it]
                 all          11          37       0.799       0.216       0.199       0.122
```

Fig. 45: mAP$_{50}$ on 88 without data augmentation with YOLOv7 model



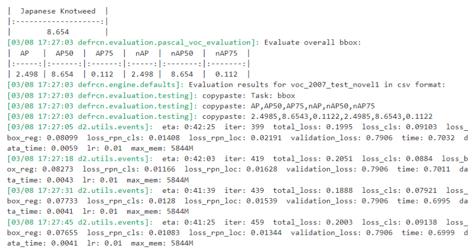
```
Scanning '/content/data/val/labels' images and labels... 11 found, 0 missing, 0 empty, 0 corrupted: 100% 11/11 [00:00<00:00, 1812.
New cache created: /content/data/val/labels.cache
               Class      Images      Labels           P           R      mAP@.5  mAP@.5:.95: 100% 1/1 [00:02<00:00,  2.41s/it]
                 all          11          37       0.667       0.378       0.326       0.141
```

Fig. 46: mAP$_{50}$ on 88 shot with data augmentation with YOLOv7 model



```
#background addtion
import os

image_dir = "/content/file"
image_files = os.listdir(image_dir)

def get_dataset_dicts():
  dataset_dicts = []

  for idx, image_file in enumerate(image_files):
      # Construct the file path to the image
      image_path = os.path.join(image_dir, image_file)

      # Create a dictionary to represent the image and its metadata
      record = {
          "file_name": image_path,
          "image_id": idx,
          "height": 800,
          "width": 1333,
          "annotations": []
      }

      # Append the record to the list of dataset dictionaries
      dataset_dicts.append(record)
  return dataset_dicts

DatasetCatalog.register("bckg", lambda: get_dataset_dicts())
MetadataCatalog.get("bckg").set(thing_classes=[])
```

Fig. 47: Background code fragment in Faster RCNN model



```
|  Japanese Knotweed  |
|:-------------------:|
|        8.654        |
[03/08 17:27:03 defrcn.evaluation.pascal_voc_evaluation]: Evaluate overall bbox:
|  AP   |  AP50  |  AP75  |  nAP  |  nAP50  |  nAP75  |
|:-----:|:------:|:------:|:-----:|:-------:|:-------:|
| 2.498 | 8.654  | 0.112  | 2.498 |  8.654  |  0.112  |
[03/08 17:27:03 defrcn.engine.defaults]: Evaluation results for voc_2007_test_novel1 in csv format:
[03/08 17:27:03 defrcn.evaluation.testing]: copypaste: Task: bbox
[03/08 17:27:03 defrcn.evaluation.testing]: copypaste: AP,AP50,AP75,nAP,nAP50,nAP75
[03/08 17:27:03 defrcn.evaluation.testing]: copypaste: 2.4985,8.6543,0.1122,2.4985,8.6543,0.1122
[03/08 17:27:05 d2.utils.events]:  eta: 0:42:25  iter: 399  total_loss: 0.1995  loss_cls: 0.09103  loss_
box_reg: 0.08099  loss_rpn_cls: 0.01408  loss_rpn_loc: 0.02191  validation_loss: 0.7906  time: 0.7032  d
ata_time: 0.0059  lr: 0.01  max_mem: 5844M
[03/08 17:27:18 d2.utils.events]:  eta: 0:42:03  iter: 419  total_loss: 0.2051  loss_cls: 0.0884  loss_b
ox_reg: 0.08273  loss_rpn_cls: 0.01166  loss_rpn_loc: 0.01628  validation_loss: 0.7906  time: 0.7011  da
ta_time: 0.0043  lr: 0.01  max_mem: 5844M
[03/08 17:27:31 d2.utils.events]:  eta: 0:41:39  iter: 439  total_loss: 0.1888  loss_cls: 0.07921  loss_
box_reg: 0.07733  loss_rpn_cls: 0.0128  loss_rpn_loc: 0.01539  validation_loss: 0.7906  time: 0.6995  da
ta_time: 0.0041  lr: 0.01  max_mem: 5844M
[03/08 17:27:45 d2.utils.events]:  eta: 0:41:25  iter: 459  total_loss: 0.2003  loss_cls: 0.09138  loss_
box_reg: 0.07655  loss_rpn_cls: 0.01083  loss_rpn_loc: 0.01344  validation_loss: 0.7906  time: 0.6999  d
ata_time: 0.0041  lr: 0.01  max_mem: 5844M
```

Fig. 48: Validation loss logging for 10 shot seed 0 by DeFRCN model

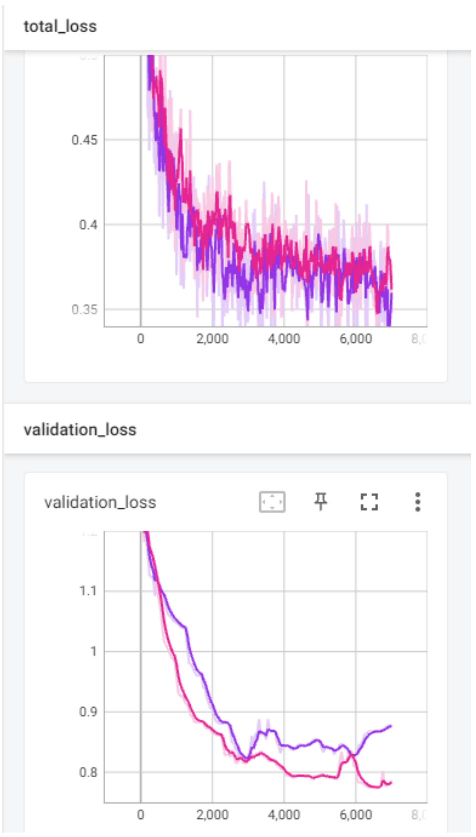


Fig. 49: Validation loss curve of 10 shot on seed 0 with learning rate of 1xe^{-4} by DeFRCN model. The pink line is with data augmentation and the purple is without data augmentation.

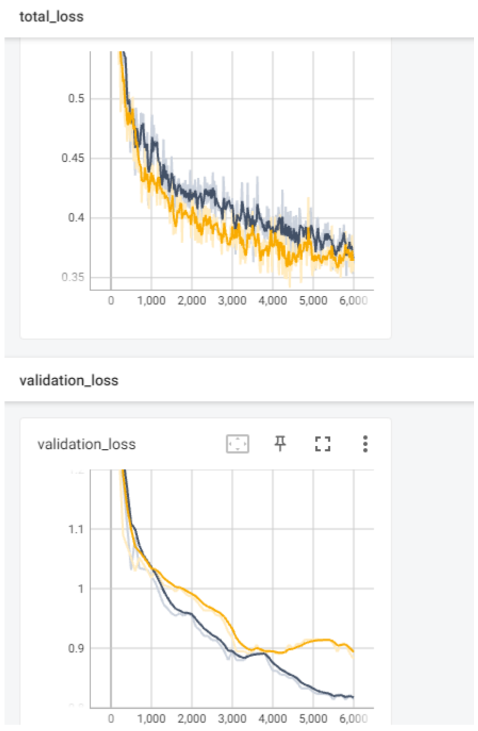


Fig. 50: Validation loss curve of 5 shot on seed 0 with learning rate of 1xe^{-4} by DeFRCN model. The black line is with data augmentation and the yellow is without data augmentation.

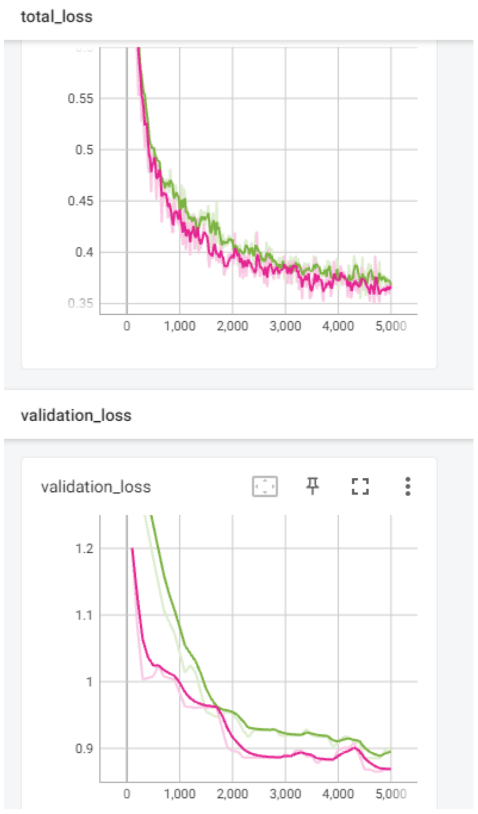


Fig. 51: Validation loss curve of 3 shot on seed 0 with learning rate of 1xe^{-4} by DeFRCN model. The pink line is with data augmentation and the green is without data augmentation.

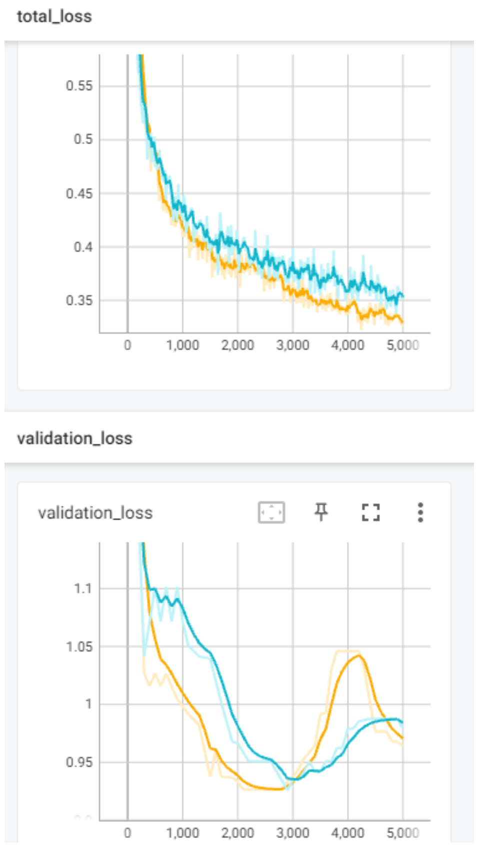


Fig. 52: Validation loss curve of 2 shot on seed 0 with learning rate of 1xe^{-4} by DeFRCN model. The blue line is with data augmentation and the yellow is without data augmentation.

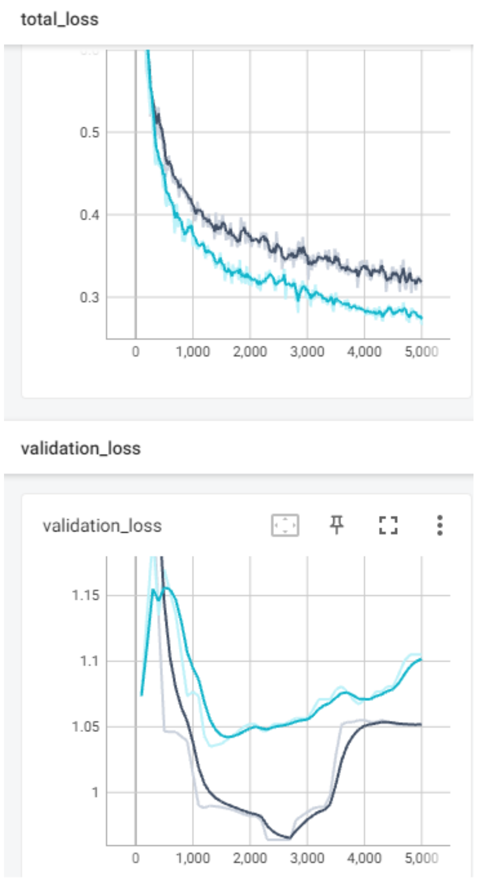


Fig. 53: Validation loss curve of 1 shot on seed 0 with learning rate of $1\text{x}e^{-4}$ by DeFRCN model. The black line is with data augmentation and the blue is without data augmentation.

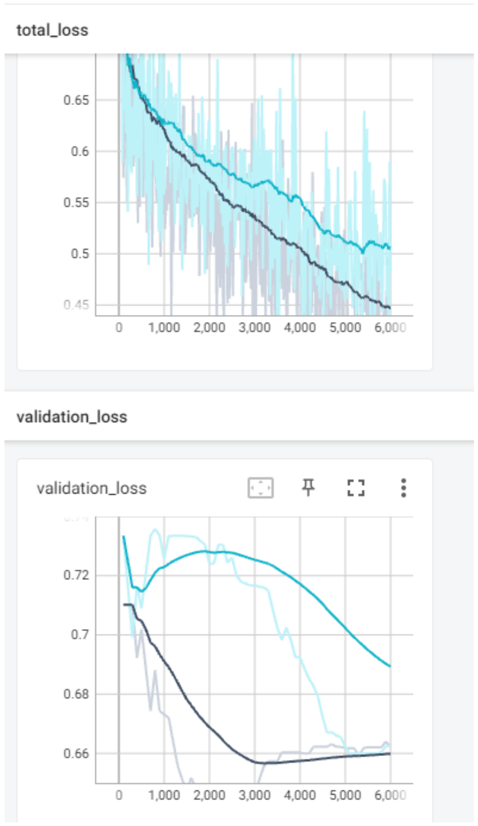


Fig. 54: Validation loss curve of 88 shot on seed 0 with learning rate of $1\text{x}e^{-4}$