Improving Robustness of Image Recognition Through Artificial Image Augmentation

A thesis

submitted in fulfilment

of the requirements for the degree

of

Master of Engineering

at

The University of Waikato

by

Callum Herbert

THE UNIVERSITY OF
WAIKATO
Te Whare Wānanga o Waikato

2023

Abstract

Deep learning based computer vision technologies can offer a number of advantages over manual labour inspection methods such as reduced operational costs and efficiency improvements. However, they are known to be unreliable in certain situations, especially when input images contain augmentations such as occlusion or distortion that computer vision models have not been trained on. While augmentations can be mitigated by controlling some situations, this is not always possible, especially in outdoor environments.

To address this issue, one common approach is supplemental robustness training using augmented training data, which involves training models on images containing the expected augmentations to improve performance. However, this approach requires collection of a substantial volume of augmented images for each expected augmentation, making it time-consuming and costly depending on the difficulty involved in reproducing each augmentation.

This thesis explores the viability of using artificially rendered augmentations on unaugmented images as a substitute for the manual collection and preparation of naturally augmented data for image recognition and object detection models. Specifically, this thesis recreates nine environmental augmentations that commonly occur within outdoor environments and evaluates their impact on model performance on three datasets.

The findings of this thesis indicate potential for using artificially generated augmentations as substitutes for naturally occurring augmentations. It is anticipated that further research in this area will enable more reliable image recognition and object detection in less controllable environments, thus improving the results of these technologies in uncertain situations.

Acknowledgements

I would like to express my deepest regards to my supervisors, Dr. Panos Patros, Dr. Michael Mayo, Assoc. Prof. Melanie Ooi, and Assoc. Prof. Judy Bowen. For their guidance and insights throughout the research process.

I would like to further acknowledge both Assoc. Prof. Melanie Ooi for her support, passion and selfless involvement in this project, and Assoc. Prof. Judy Bowen, who took over as my primary supervisor near the end of the project, requiring her to quickly familiarize herself with the research within a limited time frame.

The research conducted in this thesis relies on the work completed by a significant number of people. I would like to extend my appreciation to the members of the DeepWeeds project, the Wellington Camera Trap project, the Bio-Heritage project, and those involved in annotating the Open Images V6 dataset.

I would also like to thank my employer and co-workers for their support throughout this project. Their understanding and cooperation enabled me to fully commit to this research by shouldering the additional workload my absence caused.

Finally, I would like to extend my sincere gratitude to my family for their support throughout the duration of this research.

Table of Contents

Table of Figures

Table of Tables

Table of Equations

1. Introduction

Computer vision is a sub-field of artificial intelligence that aims to replicate some elements of human vision. It has become a critical component in many industries as a result of the cost and efficiency advantages it provides through enhancement to manual labour inspection methods. By enabling the completion of tasks with greater efficiency and without being subject to fatigue, illness or distractions, the use of computer vision both frees up workers for more complex tasks that cannot yet be automated and enhances processes within existing workforces. Some existing applications of computer vision in industry include harvesting of tea leaves [1] and freight measurement [2].

Despite the advantages of computer vision, manual quality control checks are often implemented in conjunction with computer vision due to its unreliability in situations where uncontrolled events are expected to occur, such as in the event of a hardware failure or a change in environment [3, 4]. In many cases, unreliability can be attributed to an overdependence on the image features used to identify an object which results in poor generalisation when the computer vision system encounters unexpected situations. This thesis aims to reduce this unreliability.

Humans possess an innate ability to derive context from abstract images, a skill not yet fully replicated by any computer vision framework but is under active improvement as shown through advancements in attention awareness [5], image-to-image translation [6, 7] and adversarial learning [8]. These advancements are not the focus of this thesis but help to indicate some of the range of work involved in replicating human vision. Most importantly, advancements in computer vision are becoming more commonly applied to real-world processes, for example in the medical sector [9-12], suggesting this is a desirable topic and will likely be integrated into other sectors and industries as its popularity increases. Through exploration of improved reliability, this thesis hopes to assist computer vision's continual integration into real-world processes.

Computer vision comprises several fields of artificial intelligence-based technologies, with image recognition and object detection being two of the most notable. While these fields share some similarities, such as their usage of deep learning methods to provide context for an image, they differ in terms of their goals. Image recognition aims to classify an image based on the

context of the whole image, while object detection is used to identity one or more objects within an image. In the context of replicating the abilities of a person, image recognition answers the question *"What is this object?"* whereas object detection answers *"What objects can be seen here?"*. Both fields have been of interest to both researchers and industry over the last decade as is evident through the volume of research and adoption by industries [13, 14].

Image recognition and object detection are achieved through specially designed programs that use one of many available image processing frameworks such as OpenCV [15], TensorFlow [16] or PyTorch [17]. These frameworks on their own cannot perform computer vision tasks but provide a structured Application Programming Interface (API) for utilising computer vision models, in particular, deep learning models. Deep learning models designed for image recognition and object detection are a type of artificial neural network that use a combination of convolutional and pooling layers designed to extract features from an input image and generate a prediction based on identified features. These frameworks also provide the ability to train models using supervised, unsupervised or reinforcement learning methods.

Supervised learning is the most common method used to train models. This method involves labelling each image by supplying either the correct classification or classification bounds. During the supervised learning training process, a model is provided with labelled training data which is comprised of objects the model is expected to be able to recognise once training has completed. Each type of object the model is trained to recognise or locate is called a class. In unsupervised learning, labels are not provided. Instead, the model determines its own classifications. Reinforcement learning also does not provide labels, however it provides feedback to the model after each prediction. It is important to note that training only teaches a model how to recognise the classes within the training data. To train a model to either recognise new classes or better recognise existing classes, additional training is required. Deep learning models are discussed in more detail in Chapter 2.

A model's performance, usually measured in the percentage of predictions correctly classified, is dependent on both the quality and variety of the training data it is trained on. A significant reduction in performance is expected when trained on low quality, blurry images as there is likely not enough detail to allow a model to positively learn the difference, for example, it is difficult to consistently distinguish between a cat's ear and a dog's ear when the quality of training images is insufficient as shown in Figure 1.1.

Cat Ear Dog Ear

Figure 1.1 Low quality comparison of a cat versus dog ear

Although not ideal, it is common that the training images used to train a model are not representative of all the possible images that the model will be asked to evaluate in a real-world environment. Often this is because the collection and labelling of real-world data is either cost or time prohibitive, resulting in model trainers utilising similar, but not identical training data from online sources. In other situations, model trainers may elect to only train a model on unaugmented real-world images to improve performance metrics, which is also not truly representative of the possible range of images from a real-world environment. This can result in a model performing quite well during training but performing badly when tasked with classifying real-world data. This is an issue experienced by all deep learning models and stems from a lack of ability to generalise [18].

Generalisation refers to a model's ability to abstractly learn image features. A model that generalises well can recognise commonalities in new unseen images without either relying on certain features present within the training-data that is not found in the unseen-data or being thrown off by features present on the unseen-data that is not found within the training-data. Variation, either through diversely sourced images or supplemental training using different datasets, helps to mitigate this issue by exposing a model to a wider variety of images with the aim of removing dataset-specific anomalies and improve overall generalisation.

Real-world data presents many challenges due to equipment failure or environmental effects which can augment or degrade an image through means such as occlusion or distortion. As a result, not only is it important for a model to be able to generalise, but it is necessary for a model to be robust and resilient when presented with an unexpected situation. A robust model

is a model that is able to return correct predictions when given images that contain augmentations or degradations. Augmentations can occur with different severities, meaning that in extreme cases, it is possible that the image could become unclassifiable. In these cases, it is not expected that the model makes the same prediction it would without the augmentation, much like with human vision. This is the problem this thesis will address through an exploration of deep learning robustness and the viability of using artificially augmented datasets to improve model generalisation.

Robustness evaluations are typically done by comparing a model's performance on both an unaugmented dataset and an augmented dataset where an augmentation is any corruption, degradation or occlusion that still permits classification. A robust model is expected to perform consistently on the two datasets by returning similar predictions resulting in similar accuracies. Ideally, the predictions returned between the two datasets would be identical, but in most cases having the augmented dataset accuracy fall within a given range, such as within 10 percentage points of the unaugmented dataset accuracy, could be considered close enough. However, there is no standardised range and is ultimately at the discretion of the people performing the evaluations. Robustness will be discussed in more detail in Section 2.2.

Augmentations are expected to occur more frequently in environments that do not have solutions set up to mitigate them. This means that images captured within an uncontrolled environment where no mitigations have been put in place are expected to be at higher risk of augmentation. By extension, outdoor environments are expected to experience more augmentations than indoor environments as indoor environments are easier to control. Naturally, data captured within uncontrolled, outdoor environments run a higher chance of becoming augmented, such as in a rural industrial automation or in vehicle automation.

As a necessity, most devices in outdoor environments are edge devices. Edge devices are typically low powered sensors, processors or storage devices that sit on the edge of a larger cloud based network [19]. Typically, an edge device performs the minimum amount of processing required in order to conserve power and offloads all other responsibilities to another node within the cloud network. Two key issues present within most outdoor environments are often the availability of essential resources such as, power or network connectivity, and the runtime required by devices on these limited resources. However, in some situations, either a device is not cloud-enabled, cannot connect to the cloud, or must act in less time than cloud communication allows which requires that processing must be done directly on the edge device.

This can present some challenges when dealing with augmented images as will be discussed in Section 2.2.

1.1. Scope

This thesis focuses on improving model robustness on edge devices operating within uncontrolled, outdoor environments that perform either image recognition or object detection locally. It is assumed that these devices are operating on battery and as such have to minimise their energy utilisation and cannot accommodate additional processing tasks. Model improvement is achieved through the creation of a bulk image augmentation generator that can be configured to create augmented datasets for supplemental training of existing deep learning models, improving their ability to generalise.

1.2. Contributions

This thesis makes the following contributions:

- **Investigation of artificially augmented images for improving model robustness**
  This thesis demonstrates the viability of applying artificially generated augmentations to existing datasets as an alternative to the manual collection of naturally augmented datasets for supplemental model training.

- **A collection of augmentation "effects"**
  This thesis artificially replicates a small collection of real-world environmental effects and evaluates their viability for augmented dataset creation.

- **"Image Augmentation Generator" software**
  This contribution is a software utility which assists with the bulk generation of augmented training data. This utility can be run in either an interactive mode where users can experiment and configure augmentations or in a non-interactive mode which

is designed for the bulk processing of multiple images based on provided augmentation pre-sets.

This utility is not limited to the augmentations discussed in this paper and can be programmed through "plugins" to generate any number of augmentations using any desired framework.

This utility is available from https://github.com/cmherbert/Artificial-Image-Augmentation-Generator.

1.3. Structure

In Chapter 2, the background of deep learning, robustness, and augmentations is explored. This chapter provides the technical information required for the rest of the thesis and is also where this project's augmentations are determined.

In Chapter 3, the datasets used in this thesis are introduced and used to create five models per dataset that provide the baseline results which are used later in this thesis for each dataset. This chapter describes how the datasets are prepared for this project.

In Chapter 4, the creation of nine chosen augmentations is discussed in addition to the methodologies behind the generation of the augmented datasets.

In Chapter 5, the evaluations for this thesis are performed and discussed. Three evaluations are performed, each based on a question posed in Chapter 5.

Finally, Chapter 6 summaries this thesis and explores identified limitations and future work.

2. Background

This chapter begins with a review of deep learning in the context of image recognition and object detection (Section 2.1) followed by an introduction to robustness and its importance (Section 2.2). This is followed by a discussion of data augmentation (Section 2.3), including an exploration of existing augmented datasets and augmented data generators. The final section, Section 2.4 summarises key points from this chapter and describes how these form the basis of the rest of the project.

2.1. Deep Learning

Modern image recognition (IR) and object detection (OD) platforms utilise artificial neural network (ANN) models as a core component of their respective platforms. ANNs, are inspired by the brain [20], specifically aiming to replicate a cluster of biological neurons. Neurons are designed such that they can accept a number of inputs that the neuron translates into a single output, within an artificial neuron.

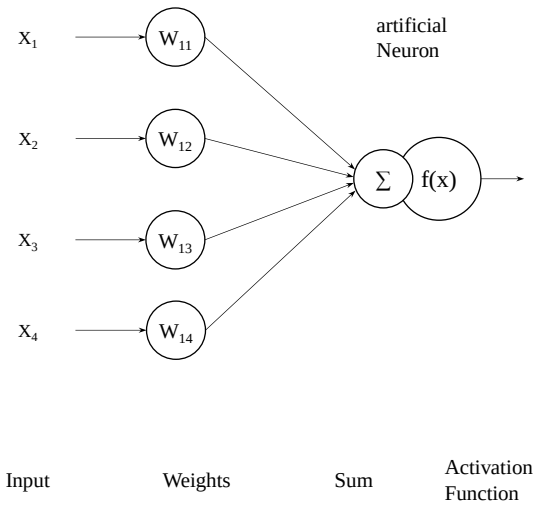


Figure 2.1 Structure of an artificial neuron. Adapted from [21] (CC0 1.0)

Figure 2.1 represents a typical artificial neuron, which shows a particular neuron receiving four inputs. Each input is weighted which increases or decreases the value provided by its attached input, serving as a means of prioritising, or strengthening inputs. These inputs are summed and passed into an activation function which is used to determine the neuron's final output. The final output is either 0 or a 1 depending on whether the result of the activation function exceeds a particular threshold value.

ANNs combine multiple layers of interconnected neurons to form the basis of a neural network. Each neuron within a layer receives inputs from every neuron in the preceding layer and provides its output to every neuron in the succeeding layer as shown in Figure 2.2.

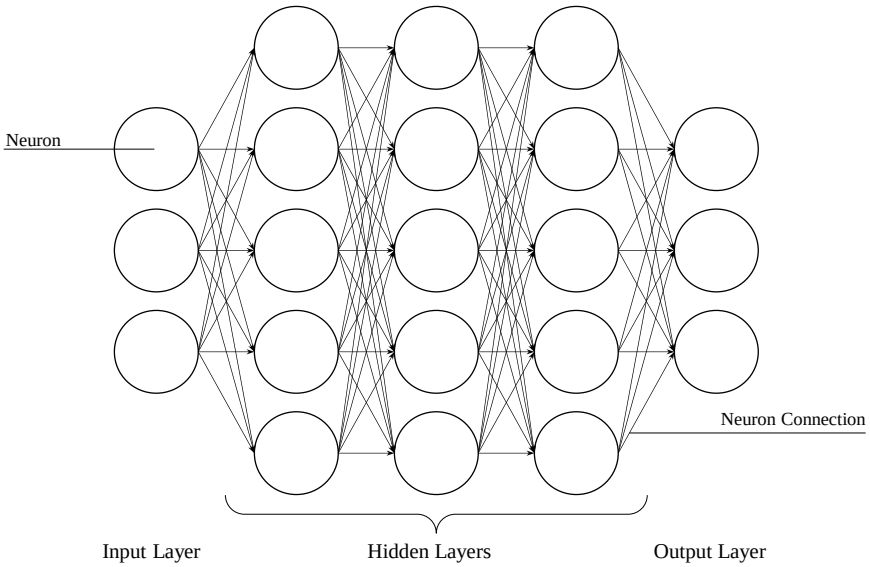


Figure 2.2 Artificial neural network structure

There is no restriction to either the number of hidden layers an ANN can have or the number of neurons within any hidden layer. The number of neurons contained by each input and output layer are typically dependant on the purpose of the ANN. In the case of IR, it is typical for there to be one neuron in the output layer for each possible classification that can be detected, however, the number of input neurons depends on how the input image pixels are prepared, for example, input images can be provided to an ANN as either a collection of pixels or a collection of the colour channels within each image.

With each additional hidden layer or neuron, ANNs become increasingly more complex and the underlying operation less clear. For this reason, ANNs are typically described as a '*black box*' as there are typically too many neural connections creating too many pathways between the input layer and output layer for any form of manual tracing. This can make ANNs hard to debug or configure manually [22, 23]. This also makes ANNs difficult to visualise, although work has been done to make this easier [24-26].

ANNs are configured by adjusting the weight values within each neuron to influence an ANN as a whole to produce desired outputs based on given inputs [27]. Abstractly, this helps an ANN recognise patterns by prioritising or un-prioritising particular pathways, with the theory that similar regions of an image are likely to activate the same pathways. Weights are configured within an ANN through training. Much like with a brain, ANNs are designed to learn how to process input rather than being explicitly programmed. An ANN on its own cannot train itself and relies on an external program to configure neuron weights. In the case of IR and OD, frameworks assist that provide the necessary functionality to perform these adjustments – some of these frameworks are discussed in sections 2.1.1 and 2.1.2. There are a few different methods these frameworks can use to train ANNs [28]:

- **Supervised Learning**

  In supervised learning, an ANN is provided with labels for each input. Each label contains the expected output. During the learning process, the framework attempts to adjust weights within an ANN to match the labels as closely as possible.

- **Unsupervised Learning**

  In unsupervised learning, no labels are used, and the framework tries to find commonalities within each input. These commonalities are grouped and result in inputs being grouped based on similarities identified by the ANN which are not necessarily the groups that are expected from supervised learning.

- **Reinforcement Learning**

  In reinforcement learning, a framework adjusts weights within an ANN and then makes further adjustments based on feedback it receives. The feedback determines whether or not the adjustments made by the framework allowed the ANN to better produce the expected output for that particular input.

Both IR and OD utilise the terminology ‘class’ to define what type of image has been recognised or object found. Classes are effectively a collection of objects which IR and OD are trained to identify where each object is a class.

ANN is an umbrella term that encompasses all types of artificial neural networks. Naturally, some ANNs with specific properties are typically classified together forming sub-types. Although in many cases ANN can correctly be used to identify a sub-type of ANN, most IR and OD frameworks refer to ANNs as models.

Deep neural networks (DNN)s are a sub-type of ANN that consists of at least two hidden layers. Multiple hidden layers allow for more interconnected neurons providing additional depth for the network. This additional depth allows DNNs to learn more abstract features from training data leading to better performance in more complex tasks such as speech and image recognition [29].

Convolutional neural networks (CNN)s are a specific type of DNN designed for image recognition which utilise a process called convolution. Convolution is used to extract features from input images. CNNs must contain at least one convolutional layer.

CNNs are specifically designed to reduce the complexity of working with computer vision data. Images can contain informational patterns that are stretched or translated anywhere within an image, something that the human eye can easily handle but an ANN not so well, as they are reliant on consistent patterns. Convolutional layers are specialised layers that break down an image into smaller sub-images or *‘feature maps’*. Feature maps still utilise neurons, (as discussed earlier), however, the neurons are trained to recognise the pixels within the feature map rather than the whole image. Additional layers can be added to create additional, but smaller feature maps.

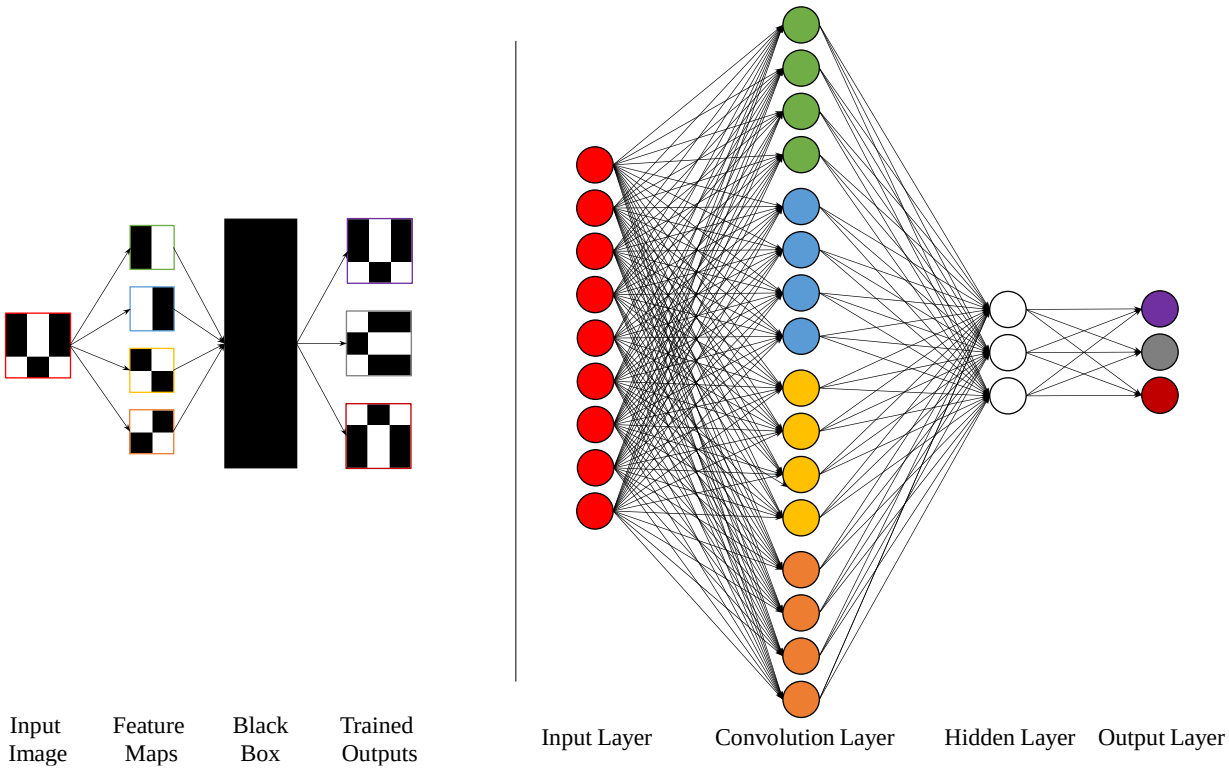


Figure 2.3 Example of a simplistic CNN showing feature map break down on left and ANN structure on right

Figure 2.3 shows an example of a simple CNN. This model is designed to identify three images representing glyphs ‘v’, ‘c’ and ‘n’ as accented in purple, grey and red respectively in the right-hand side of Figure 2.3. The left-hand side shows a 3x3 pixel image of ‘v’ being broken down into four feature maps which are used to determine the glyph.

For bigger images, convolutional layers can be followed by a pooling layer which further reduces the number of neurons required for a feature map by effectively shrinking it.

Although in many cases, the human brain needs repeated exposure to facilitate the learning process – hence the phrase, *“Practice makes perfect”*, models require much more exposure, through large datasets, to learning material during their learning process [30]. This is often a hurdle when it comes to utilising IR or OD as for each class the underlying model needs to recognise, a large collection of significantly different training images must be sought. If there are insufficient training images, the model may not properly learn all the patterns common to each input class resulting in reduced accuracy [31, 32]. If the images are too similar, overfitting can occur making the model struggle to find patterns within new dissimilar images containing the same classes it has learnt [32]. Fortunately, there exists a number of publicly available

training images for a variety of classes on the internet, these images are often designed specifically as training data ensuring quality, quantity, and diversity within each dataset. Table 2.1 gives a list of common datasets. If a dataset containing a specific class does not already exist, training data for this class will have to be manually sourced, which is a time-consuming process, as is the creation of labels for these images when used under supervised learning.

Table 2.1 List of common image datasets

Dataset	**Classes**	**Description**
Coco [33]	80	More than 200,000 labelled images with multiple objects instances, suitable for object detection
MNIST [34]	10	70,000 black and white, labelled images, each with a single handwritten digit (0-9). Suitable for image recognition
ImageNet [35]	1,000+	Growing collection of labelled images sorted into progressively narrowing categories (i.e., Vehicle -> Craft -> Watercraft -> Sailing Vessel -> Sailboat -> Trimaran). As this dataset is being added to, subsets of this collection are often released as their own datasets, for example ILSVRC [36]
PASCAL [37]	20	Designed as a competition dataset growing in both number of classes and images between 2005 and 2012. PASVAL VOC 2012 contains 11,530 images and is suitable for object detection [38]
CIFAR [39]	10, 100	Comes in two versions, CIFAR-10 which both contain 60,000 labelled images. CIFAR-10 contains 6,000 images per class whereas CIFAR-100 contains only 600. Suitable for image recognition. [40]
Open Images [41]	OD: 600 IR: 20,638	A large collection of labelled images. Approximately 1.9 million of these images are suitable for object detection and 61 million suitable for image recognition.

In addition to training data, most datasets also contain validation and test data. These three categories are made of similar, but unique images that allow a model to be analysed independently of the data it was trained on. During training, a model is trained using the training data, and the resultant model is validated using the validation data. This is important as IR and OD frameworks provide the tools that enabled performance tacking while training. For example, if while training, a model continually performs well on the training data and poorly on the validation data, this is a sign of overfitting. The test data is not involved in training at all and is solely used for evaluating performance of a model outside training. Performance of a model is typically measured by determining its accuracy, although in some cases, especially when speed is preferred, model performance is evaluated through the time it takes to evaluate an image. Accuracy calculations require labels for each image such that a prediction generated by a model can be compared to something true. As a result, the accuracy calculation is simply a comparison between the models' correct predictions and the correct classifications for each image as determined by the image labels.

2.1.1. Image Recognition

Image recognition is utilised within many deep learning applications that require trained neural networks to classify images. Neural network models are typically trained using supervised learning. This involves a large dataset of labelled images which are used to teach a model to recognise patterns and features that are common across image classes.

Most models are evaluated using their top-1 accuracy metric. Top-1 accuracy is effectively a measure of conventional accuracy, where the percentage score indicates the percentage of images correctly classified by the model. For each prediction, a model yields a confidence percentage for each possible class. To be considered correctly classified, the confidence percentage for the actual class must be higher than of all other classes within the prediction. A variation on this, known as top-5 accuracy is also used which considers a prediction correct if the actual class confidence is within the top-5 confidence percentages for the prediction. Top-5 accuracy is not particularly useful for models which have fewer than 5 classes.

Table 2.2 lists some of the most common deep learning models that have been made available over the last decade showing the improvement in their top-1 accuracies on similar variants of the ImageNet dataset.

Table 2.2 List of some image recognition models with their top-1 accuracy based on the ImageNet dataset

Model	**Year**	**Top-1 Accuracy**	**ImageNet version**
AlexNet [42]	2012	59.3%	ILSVRC-2010
VGG16 [43]	2014	75.3%	ILSVRC-2014
Inception-V1 (GoogLeNet) [44]	2014	68.3%	IMAGENET-1K
ResNet-152 [45]	2015	80.6%	ILSVRC-2014
Inception-V3 [46]	2016	81.2%	ILSVRC-2012
Inception-V4 [47]	2017	82.3%	ILSVRC-2012
ResNeXt-101 [48]	2017	80.9%	IMAGENET-1K
MobileNet-V2 [49]	2018	72.0%	IMAGENET
NASNet-A [50]	2018	82.7%	IMAGENET
EfficientNet-B7 [51]	2019	84.3%	*Possibly ILSVRC-2015*
ViT-H/14 [52]	2020	88.6%	ILSVRC-2012
gMLP-B [53]	2021	81.6%	IMAGENET

Within the ImageNet dataset, images are distributed evenly among each class which means that even if a model has difficulty with a particular class, it is expected that this difficulty will be reflected by the top-score. For example, in the case of a model that is trained to detect five classes evaluates a evenly distributed dataset, it is expected that each class can contribute up to 20% of the final top-1 accuracy. However, as it is not always possible to guarantee that images will be evenly distributed among classes, a scenario can occur when a class has a relatively low representation in a dataset compared to other classes. In this situation, performance, either good or bad, is likely to be obscured behind the performance of other, better represented classes making Top-1 accuracy undesirable for real-world evaluation. This issue can be overcome by calculating a model's F1-score instead, much like top-1 accuracy, this metric is also measured as a percentage.

An F1-Score combines a model's precision and recall metrics to calculate the harmonic average of each classes' accuracy. In order to calculate an F1-score, classification predictions must be sorted into one of the four categories shown in Figure 2.4.

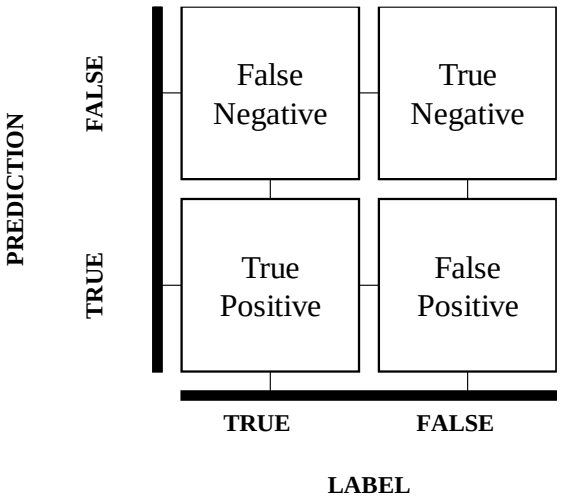


Figure 2.4 Prediction categories

All images when evaluated will fall into one of these four categories. Ideally, images would only fall into either True Positive (T_p) or True Negative (T_n) categories which indicates perfect performance. However, this is unrealistic, and mistakes are likely to be made by the model which result in a model making a classification when there is nothing to classify (False Positive, F_p) or a model making no classification where there was something to classify (False Negative, F_n). In the context of non-binary IR, a positive detection of class A when the label suggests class B is considered a false positive.

These categories are used to calculate the precision and recall metrics. Precision indicates the percentage of true positive predictions out of all positive predictions. Recall indicates the percentage of true positive predictions out of all positively labelled images.

$$P = \frac{T_p}{T_p + F_p}$$

Equation 2.1 F1-Score precision calculation

$$R = \frac{T_p}{T_p + F_n}$$

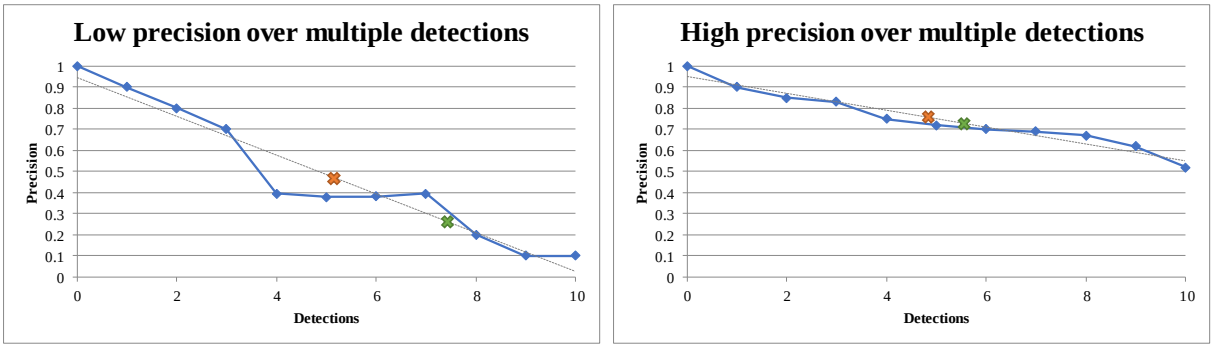
Equation 2.2 F1-Score recall calculation

$$F1 = 2 \times \frac{P \times R}{P + R}$$

Equation 2.3 F1-Score calculation

Equation 2.1, Equation 2.2 and Equation 2.3 are used to calculate each classes F1 score. Calculating the top-1 accuracy of IR models is straightforward but can become complicated when applied to scenarios containing unbalanced class distributions within the labelled dataset used for training and evaluation. The model may perform better at predicting one class over another class which can result in a biased accuracy if the model performs better on a class with more samples than a class the model performed poorly with.

F1-score calculation mitigates this issue through the consideration of both precision and recall and as a result, the F1-Score is a calculation of the harmonic mean rather than an arithmetic (traditional) mean, which creates a bias towards lower values. This can be useful then comparing precision and recall as this results in the penalisation of false negatives. In applications where a missed detection can have serious consequences such as in the case of stop sign detection in autonomous cars, F1-score can be considered a more appropriate metric than the top-1 accuracy alone.



Traditional Average Precision	0.486	✖	**Traditional Average Precision**	0.750
Harmonic Average Precision	0.274	✖	**Harmonic Average Precision**	0.727

Figure 2.5 Comparison of traditional and harmonic averaging using two graphs showing a different drop in precision over ten predictions

Figure 2.5 presents two graphs each showing a model's overall precision during the evaluation of ten predictions. The left-hand graph experiences a steeper drop in precision over the ten predictions than the right-hand graph. Two points are marked on each graph, an orange point representing the average precision of the model over the ten predictions and a green point representing the harmonic average precision. Notably, as the overall precision increases, the

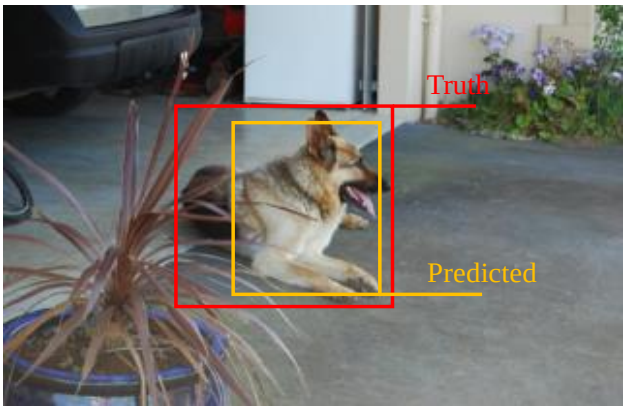
harmonic and traditional averages appear closer together. These graphs demonstrate how harmonic averaging, and by extension the F1-score penalises poorer values resulting in a better metric for use in evaluating IR performance.

2.1.2. Object Detection

Object detection is an advancement on Image Recognition where the objective is to locate and then classify objects within an image or video. Like IR, OD has been researched and improved upon for more than two decades [13]. Modern deep learning-based OD began around 2012 and since 2014 OD frameworks have fallen into one of two categories:

- **One-stage detector**
  Performs object location and classification in a single stage on the whole image. Typically, faster but less accurate than a two-stage detector making it better for real-time applications.
- **Two-stage detector**
  Performs object location and classification in two stages. In the first stage, regions of interest (ROI) are identified which are a cropping of the input image deemed likely to contain an object. These ROIs are passed onto the second stage for classification. Typically, slower but more accurate than a one-stage detector.

Determining the accuracy of OD models is more complex than calculating accuracies of IR models. Firstly, creation of labels requires not only a class for each object, but four additional values representing the region in which an object exists.



Area Truth	34,010 px
Area Predicted	19,431 px
Area Intersect	19,431 px
Area Union	34,010 px
IoU	**0.57**

Figure 2.6 Object detection truth vs. prediction with IoU calculation table

Figure 2.6 shows a picture of a dog with both the labelled truth (red) and predicted (gold) bounds. Assuming the predicted class matches the truth class, the two regions must be evaluated. The two regions are compared by computing their intersection over union (IoU). The IoU calculation indicates the percentage of overlapping area from the total areas of the truth bounds (A_t) and the predicted bounds (A_p) as shown in Equation 2.4.

$$IoU = \frac{A_t \cap A_p}{A_t \cup A_p}$$

Equation 2.4 IoU calculation

An arbitrary IoU threshold value is used to determine if a prediction is marked, as either a positive detection (true positive) or a negative detection (false positive), where a positive detection has an IoU at or greater than the IoU threshold. This means that the prediction in Figure 2.6 which has an IoU of 0.57 would be considered positive if the IoU threshold was set as 0.3 and negative if the IoU threshold was set at 0.9. Keeping with the categories defined in Section 2.1.1, a prediction made where there is no label is considered a false positive, the absence of a prediction where a label is present is a false negative.

Using the IoU calculation, all predictions can be assigned as either true positives, false positives, or false negatives. This allows the calculation of both a class's recall and precision values, however rather than using these values to calculate the F1-score as done with IR, these values are typically used to calculate a class's average precision (AP). AP represents an

approximation of the area under a precision versus recall plot whereas the F1-score computes its harmonic average, as a result, AP serves as a better metric for ongoing calculation as the F1-score simplifies the plot. Equation 2.5 shows how AP is calculated, where n is the number of plot points the plot contains.

$$AP = \sum_{i=1}^{i=n} (R_i - R_{i-1}) \times P_i$$

Equation 2.5 Average precision calculation

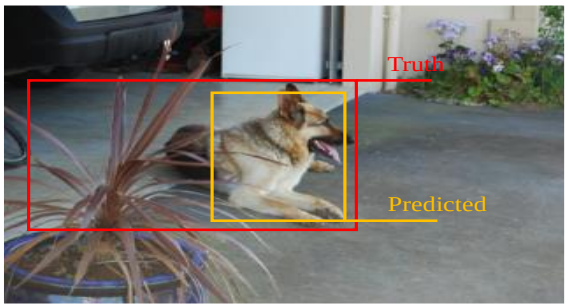
Mean average precision (mAP) represents the average AP over each class the OD model can classify and is calculated as is shown in Equation 2.6.

$$mAP = \frac{1}{n} \sum_{i=1}^{i=n} AP_i$$

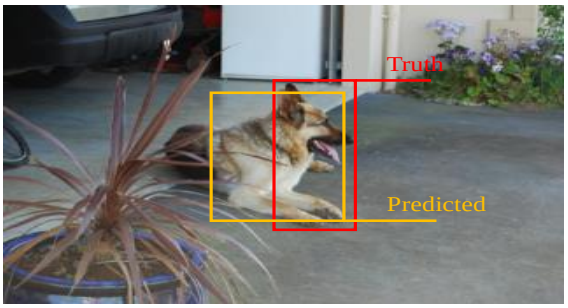
Equation 2.6 Mean average precision calculation

The IoU threshold is a significant contributor towards the calculation of a models mAP, for example, setting the IoU threshold higher can increase class precision by only classifying high-confidence classifications as positive, reducing false-positives. However, this also reduces the recall if a number of predictions which would have been classified as positive are now negative, which increases the number of false negatives. This introduces complexity when comparing mAP values of different models as each model is different and can produce different mAP results under different IoU thresholds. To provide a fairer comparison, averaging the AP for each class over ten IoU thresholds has become standard. Introduced for evaluating the COCO dataset, $mAP@[.5{:}.05{:}.95]$ averages ten IoU calculations utilising each IoU threshold from .50 through .95 with an interval of .05 [54].

Precise bounding area labelling is important for accurate OD performance evaluation as imprecise labelling can result in either valid predictions being marked as false negatives or invalid predictions being marked as true positives. This is largely a side effect of the IoU calculation where the ground truth must be assumed reliable.



Area Truth	57,280 px
Area Predicted	19,431 px
Area Intersect	19,431 px
Area Union	57,280 px
IoU	**0.34**



Area Truth	13,783 px
Area Predicted	19,431 px
Area Intersect	10,098 px
Area Union	23,116 px
IoU	**0.44**

Figure 2.7 Comparison of resultant IoU values as a result of modifying an object's truth bounds

Both tables in Figure 2.7, which shows two variants of Figure 2.6 with different truth bounds, present a decrease in IoU compared to the calculated IoU in Figure 2.6 which shows the importance of precise labelling when evaluating OD models.

Table 2.3 presents a non-exhaustive list of common object detection frameworks and either their mAP or $mAP@[.5{:}.05{:}.95]$ accuracy on the PASCAL VOC 2012 and COCO 2017 test-dev datasets respectively.

Table 2.3 List of some object detection frameworks

Model	**Type**	**Year**	**mAP**	**mAP@[0.5:0.05:.95]**	**Dataset**
R-CNN BB [55]	Two-stage	2014	53.3%	-	PASCAL VOC 2012
Fast R-CNN [56]	Two-stage	2015	68.4%	-	PASCAL VOC 2012
YOLO [57]	One-stage	2015	57.9%	-	PASCAL VOC 2012
SSD512 [58]	One-stage	2016	74.9%	-	PASCAL VOC 2012
RetinaNet [58]	One-stage	2017	-	39.1%	COCO 2017 test-dev
Mask-RCNN [59]	Two-stage	2017	-	38.2%	COCO 2017 test-dev
Cascade-RCNN [60]	Two-stage	2018	-	42.8%	COCO 2017 test-dev
YOLOv4 [61]	One-stage	2020	-	43.5%	COCO 2017 test-dev

EfficentDet-D7x [62]	One-stage	2020	-	55.1%	COCO 2017 test-dev
RepPoints V2 [63]	One-stage	2020	-	52.1%	COCO 2017 test-dev
Sparse R-CNN [64]	Two-stage	2021	-	42.3%	COCO 2017 test-dev

The COCO 2017 test-dev dataset, along with measuring the mAP over 10 intervals has become the standard over VOC 2012 since 2017 as is reflected in the table. This makes it difficult to directly compare the metrics used by the different, however the change allows for a better representation of a model's performance by considering a wider range of IoU thresholds making it much harder for a model to reach 100% average precision and hopefully will provide a better foundation for performance comparisons of future models.

2.2. Robustness

As discussed in Section 2.1, accuracy is the primary metric involved in ascertaining a model's performance. Typically, evaluations are performed on a discrete subset of the overall training-set, known as the test-set, within a training environment. In theory, this should result in a high accuracy metric due to the expected similarities between both the testing and training sets. However, it is more commonly seen that when evaluated within a real-world environment using real-world images, the accuracy is far lower than within the training environment. In the context of this thesis, a real-world environment is an uncontrolled environment that a model is intended to be deployed to.

A known problem with models is their lack of ability to generalise well on unseen data, especially when overfitted [32]. Models are reliant on patterns learned during their training, and therefore, new patterns, even if similar can be too different for a model to confidently identify. Uncontrolled environments can increase this problem as they often introduce additional complexity. When performing IR or OD it is very difficult to ensure captured images will be similar to that of the training data. Uncontrolled environments are often subject to environmental issues such as weather or lighting and technological issues such as camera failure or noise. While in some scenarios these problems can be easily mitigated, for example by using careful camera placement, this is not always possible. This means that models need to be able to handle whatever image is provided irrespective of quality.

For the rest of this thesis, all impacts on images caused by environmental or technical issues will be referred to as *'augmentations'*. An augmentation in the context of this thesis, is either a corruption, degradation or perturbation applied to, or present on, an image. If an image has no augmentations, it is considered *'clean'*. Of course, some augmentations can occur with different levels of intensity. In combination with the ambiguous nature of how models identify learned classes, this means it is not always clear how an augmentation will affect predictions. In some cases, a model might be naturally resistant, where it returns correct predictions despite the presence of a particular augmentation, while in others the slightest change to a few pixels could cause an incorrect prediction [65, 66].

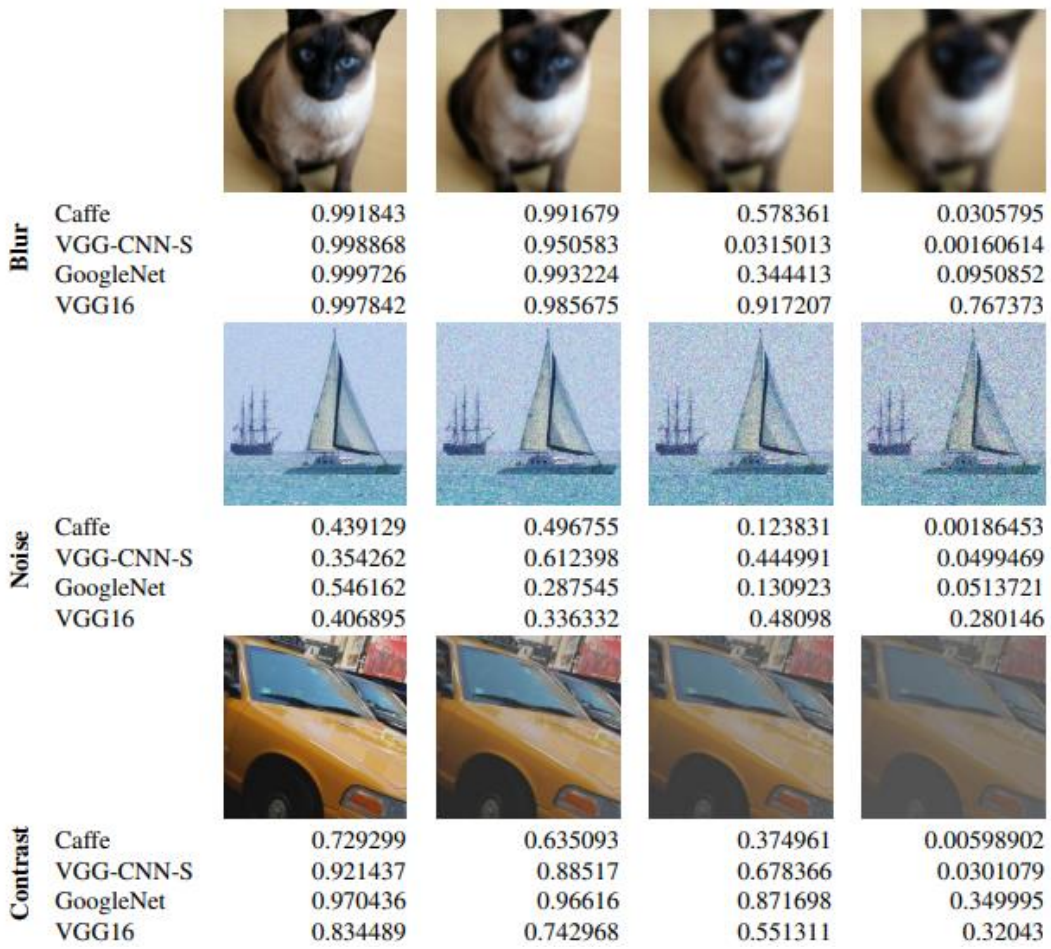


Figure 2.8 Examples of distorted images showing the confidence prediction of four different DNN models for the correct class under each image. Adapted from [67] (© 2016 IEEE)

Figure 2.8 shows the impact of three different augmentations at four different intensities for four different models. Under each image is the confidence prediction of the correct class from its respective model. Of note, while there is a consistent trend that each model performs worse from left to right as intensity increases, each model's confidence decays at a different rate for each effect. This shows that in fact, some models are naturally able to cope with an augmentation better than another augmentation and that some models are able to cope with a higher level of augmentation intensity than other models. It is worth noting that the models used to evaluate the images shown in Figure 2.8 would likely perform better at detecting some object classes before augmentation than others as a result of natural model biases, however this cannot be determined from the figure as each variant of each image is augmented.

Robustness is a measure of how well a model performs with a specific augmentation. In the case of IR, a robust model is expected to make the same classification on an augmented image as it would on a clean image. In the case of OD, a robust model is expected to locate the same class within the same visible area as would be located on a clean image. Ultimately, robustness is evaluated per augmentation. A model is considered robust when it does not experience a significant drop in accuracy when evaluated within a real-world environment and compared to accuracies evaluated in a training-environment [68, 69].

One challenge in evaluating robustness is determining what type of evaluation is appropriate for the presented problem. Evaluating a model's performance on a dataset as a whole may hide individual image classification errors that have been 'cancelled out' due to two incorrect, but opposite predictions. The alternative, however, involves evaluating each image individually which provides a more accurate representation of robustness, but is also more resource intensive and may not be significant enough to justify the additional resources required.

There are several approaches that aim to improve robustness of models. These include adversarial training, data augmentation [69, 70], image regularisation [71-73], and model correction [74]. It is important to note that while data augmentation and adversarial training can appear as very similar approaches which both aim to improve a model's generalisation ability, they differ in their intent and application and hence are discussed separately.

Adversarial training is the most widely researched of the robustness approaches discussed within this thesis. This approach uses adversarial attacks to improve the robustness of a model. An adversarial attack involves the creation of augmented data that is explicitly designed to mislead a model into generating an incorrect prediction. In most cases, adversarial attack

utilises attack images that have defects which are unnoticeable to humans but disruptive to models. The images contained within Figure 2.8 are examples of adversarial images, these images have been altered in such a way that the patterns learned during training by a model becoming increasingly unrecognisable to the model as the attack intensity increases. In adversarial training, adversarial datasets are created such that a model can be trained further with the expectation that the model can learn to generalise its already learned patterns allowing it to become more flexible when presented with new augmentations.

Generating datasets for adversarial attacks designed specifically to confound a particular model is a very complex process requiring either an understanding of how the model recognises specific patterns or a more adaptive generation approach where noise augmentation is tuned after each prediction to better confound the model.

Data augmentation involves the creation of additional data that can be used to supplement the initial training set. This approach is particularly useful in cases where it is not practical to attain additional training data or training data with sufficient differences to prevent over fitting. Although data augmentation shares the same goal as adversarial training which is to improve a model's generalisation ability, it includes a wider range of possible augmentations. These aim to introduce models to scenarios likely to be encountered within a real-world environment. This is achieved through the creation or collection of new training images which are all augmented. Much like with collection of the training-data, both the creation and collection of augmented data is a time-consuming process. In some cases, artificially applying augmentations to the training-dataset to create a new dataset can save some time depending on whether a generator for the desired augmentation exists. This is especially true when performing supervised learning, as labels can be reused so long as an artificially applied augmentation does not sufficiently occlude an image to the point that a new label is required.

Data augmentation is already utilised within many IR and OR training frameworks as a tool to mitigate overfitting within their underlying models. During training, frameworks will apply small modifications to training images such as colour adjustments and skew adjustments.

Image regularisation aims to make all input images consistent by adjusting all input images before handing over to the IR or OD framework. This includes all training, evaluation, and real-world data. By ensuring all images are consistently similar and unaugmented, no supplemental model training is required. Regularisation is used to remove specific augmentations from an image. However, within a real-world environment, this can be

problematic as it cannot be guaranteed that any particular image will contain a known augmentation or that an image will contain a single augmentation.

One possible solution to this would be through training another OD component to screen all input images and identify augmentations, something that has already been investigated [75]. The image regularisation component could then use this added context to properly regularise the problematic image.

Model correction is a lesser used and more complex approach than the three approaches already discussed. Model correction is specifically intended for image processing, as it involves modifying a model by adding *'correction units'* to any hidden layer which is identified as being influenceable by augmentation. A correction unit is effectively a collection of additional trainable convolutional layers which appear to significantly improve a model's vulnerability to the impact of image augmentation.

Considering the scope of this thesis which includes uncontrolled edge devices in an outdoor environment, the focus is on robustness approaches that are applicable to low-powered, remote devices. Image regularisation is not a desirable solution for this due to the additional processing required on each image captured prior to performing IR / OD. Additional processing on low-powered devices adds processing time and power consumption, both of which are likely to be at a premium depending on the rate of image capture and power availability. Model correction is a promising approach, however, arguably too complex for implementation by industrial developers who are assumed to be adept at utilisation of IR / OD frameworks, but not with manipulation of neural networks.

The two remaining approaches, data augmentation and adversarial training, as noted earlier, both aim to improve a model's ability to generalise, however do so using slightly different applications and with different intents. Both approaches suit edge devices, as the only changes made are in the form of tuning already-present weights within a model. However, as adversarial training requires either some level of knowledge of how a model performs pattern recognition or commitment to a complex adaptive adversarial attack, data augmentation serves as a simpler approach better suited to the project described in this thesis.

Overall, there are a multitude of viable approaches for improving model robustness, each having their own advantages and disadvantages making different approaches more desirable for different use cases. Data augmentation is the best of the four discussed approaches for this project as it provides robustness without additional runtime-resources. This means that an edge

device is not required to perform additional computation or processing tasks during either IR or OD. Data augmentation also better suits the intent of this thesis, which is not to teach a model to better generalise through material designed to make a model fail, but to teach a model to better generalise through supplemental exposure to augmented imagery.

Furthermore, looking beyond the scope of this project, work done within the realm of data augmentation is far easier to expand to new projects with different scopes as there is no reliance on a particular model's implementation or weakness. The data augmentations discussed in the next section can be applied to any model, so long as the augmentation is contextually relevant within the models anticipated real-world environment.

2.3. Augmentations

There are a number of potential augmentations that could be encountered within the given scope which are explored within this section. Section 2.3.1 explores pre-augmented datasets. Section 2.3.2 explores utilities for the generation of augmentations and 2.3.3 attempts to determine which augmentations from 2.3.1 and 2.3.2 are worth additional exploration.

Finally, in Section 2.3.4 a collection of nine augmentations are proposed to be carried forward through the rest of this thesis.

2.3.1. Augmented Datasets

Augmented datasets are a collection of training, validation, and test augmented images. There is no reason a dataset cannot contain images with different augmentations or images with multiple augmentations. However, in general it is far easier to compare the accuracy results of consistently clean data to the accuracy results of images contain a consistent augmentation.

Clean image from
training-environment

Augmented image from
real-world environment

Figure 2.9 Visual reference for discussion on real-world vs. training environment comparisons

Ideally, all training would be conducted on data from within the model's real-world environment. However, due to the difficulty of collecting an adequate volume of clean, consistent training images it is typical for training data to be sourced from an online dataset. Figure 2.9 presents two images, each an image of a different tree. If a hypothetical model's accuracy were to be compared between these two images, it would be an unfair comparison as the underlying tree image is different and as a result, it is difficult to determine whether the base image, augmentation or a combination of both factors caused a difference in accuracy. Hence performing robustness evaluation on augmented versions of the training-environment's dataset is preferred.

Creation of augmented versions of training datasets can be achieved through digital manipulation. This presents a few challenges, such as creating unique augmentation patterns that are not just a single pattern applied consistently to multiple images, and creating realistic replications of augmentations. Additionally, in the case of some augmentations that provide occlusion when performing supervised learning, a significant amount of time may be required to manually inspect if each image's label is still correct for the augmented image. Table 2.4 presents a collection of augmentations found within the MNIST-C [76], ImageNet-C [69, 77] and CIFAR-10-C [69, 77] datasets

Table 2.4 Table of augmentations and the datasets from which they can be sourced

Augmentations		
Shot Noise	Zigzag	Frost
Impulse Noise	Canny Edges	Contrast
Glass Blur	Gaussian Noise	Elastic
Motion Blur	Short Noise	Pixelate
Shear	Defocus Blur	JPEG
Scale	Frosted Glass Blur	Translate
Rotate	Zoom Blur	Stripe
Brightness	Snow	Fog
Spatter	Dotted Line	

Although as already discussed, augmented data without a clean reference is hard to fairly evaluate, a clean reference is not required for a model to benefit from augmented dataset training. There also exist datasets which contain real-world, non-artificial augmentations that can be used to improve a model's robustness towards particular augmentations such as the universal face detection dataset (UFDD), which is a combination of FDDB, WIDER FACE, IJB-C and UCCS datasets providing naturally occurring rain, snow, haze, illumination, blur, lens impediments and adversarial animal faces (distractors which look similar to human faces but actually animal faces) [78, 79].

2.3.2. Augmented Data Generators

Generation of augmentations is fairly trivial in most cases as there already exists a number of utilities that support their creation, such as Adobe Photoshop. Photoshop allows *'effects'* to be added to images such as blur, noise, and contrast [80]. Although some of these effects simulate realistic real-world augmentations, the majority of effects are not intended for training purposes. Significant disadvantages of the use of Photoshop are the requirement of a license to utilise the software and the fact that many available effects do not match the type of real-world augmentations expected in this project.

Open-source generators are programmable, which is a useful trait for anyone creating augmentations as this allows for much easier configuration of customised augmentations. Table 2.5 presents small, non-exhaustive list of augmentation generators.

Table 2.5 List of Augmentation Generators

Name	**Augmentation(s)**
From rain generation to rain removal [81]	Rain
Semi-supervised video deraining with dynamical rain generator [82]	Rain
Closing the loop: Joint rain generation and removal via disentangled image translation [83]	Rain
Let's Get Dirty: GAN Based Data Augmentation for Camera Lens Soiling Detection in Autonomous Driving [84]	Dirt
Benchmarking Robustness in Object Detection: Autonomous Driving when Winter is Coming [85]	Gaussian Noise Shot Noise Impulse Noise Defocus Blur Glass Blur Motion Blur Zoom Blur Snow Frost Fog Brightness Contrast Elastic Transform Pixelate JPEG Compression
Augmentor: An Image Augmentation Library for Machine Learning [86]	Elastic Distortion Perspective Rotation Shearing Cropping Erasure

Table 2.5 only lists generators found within online research papers, many more augmentation generators can be found hosted on code sharing sites such as GitHub. This restricts the number of reportable generators as many research papers opt for manual generation techniques, such as manually adding rain onto an image [87]. Much like with Adobe Photoshop effects, the intention of creators who publish augmentation generators online may not result in augmentations that match real-world augmentations expected by this project, suggesting that not all published generators are valid candidates for generating real-world datasets for model robustness training.

As discussed in Section 2.2, some IR and OD frameworks apply random augmentations as a part of their training pipelines. These augmentations tend to be simple and can be efficiently generated to help reduce overfitting without adding too much additional processing time to the overall training process. The necessity for speed and need for contextually relevant augmentations for a particular use case prevents many of the augmentations listed in Table 2.5 from becoming viable candidates for integration into IR and OD frameworks. This makes using augmented data generators for dataset creation (which expands the total effective dataset size) the preferred choice while attempting to improve a model's robustness. This has the added benefit of allowing augmented data to be further augmented using pre-existing, simpler framework integrated augmentations that already exist.

2.3.3. Augmentations in the context of outdoor edge devices

So far, a number of augmented datasets and generators have been described. However, not all of these augmentations are relevant to an uncontrolled, outdoor environment. The augmentations identified can be generally divided into one of two categories:

- **Sensor Augmentation**
  An augmentation that originates from the camera, such as noise, JPEG compression or defocused blurring. These augmentations are not actually present within the real-world environment but are added by the camera.

- **Environmental Augmentation**

An augmentation that originates from the environment, typically caused by weather or lighting variations. These augmentations are present within the real-world environment, either atmospherically or on the physical camera lens.

In general, most sensor augmentations are easy to generate with Gaussian or salt-and-pepper noise, brightness and motion blur being common examples found in Table 2.5 These augmentations can often occur as a result of camera hardware making them sensor augmentations. Based on this, while the listed augmentations are contextually relevant, they are also abundantly available for usage in creating augmented datasets so are not worth additional exploration. However, there are some augmentations that could be caused both by the environment and a camera, for example, brightness caused by external lighting and motion blur caused by wind, that should be explored from the perspective of an environmental augmentation rather than a sensor augmentation.

Weather augmentations are expected for outdoor devices. Weather can have a significant impact on image quality such as by darkening images in overcast lighting, refracting images in the presence of rain, blurring images in the presence of strong wind and obstructing points of interest in the presence of fog / haze. Naturally, all weather effects are applicable within this context.

There are some situations, such as with cloud-enabled autonomous driving cars and automated fruit pickers where edge devices are mobile. Depending on the speed of the device, the camera's target direction and the quality of the camera used, this can introduce motion blur, an effect also likely to be seen by stationary devices in strong winds.

Lighting augmentations such as sun glare and temperature augmentations are also contextually relevant as lighting can have a significant impact on the quality of an image. Too much can result in sensor saturation, creating effectively "empty" regions in an image (all white or all dark), for example, sun glare can render parts of images as "empty".

Augmentations in the form of transformations such as rotation or scaling are also relevant to this context as it cannot be assumed an object within an image will be a consistent distance from a camera or at a consistent rotation. However, as discussed in Section 2.2, transform augmentations are already utilised in most IR and OD frameworks during training to prevent overfitting.

One augmentation not found in either of the above sections is dirt. Dirt is a particularly relevant augmentation easily applicable to the scope of this project as many edge devices are likely susceptible to either wet mud or fine dust occlusions due to the nature of outdoor exposure. Unlike the augmentations discussed so far, dirt can easily be classified as an occlusion which tends to remove information from an image rather than distort it. If done without care, adding occlusion will not help improve a model's robustness. Nonetheless if applied carefully, it can serve as a useful tool to block partial patterns within an image, improving a model's ability to generalise by removing reliance on groups of patterns. Based on this, it is proposed that dirt be considered as a contextually relevant pattern going forward.

2.3.4. Discussion

All augmentations so far that have been deemed contextually relevant originate from non-technical origins. This presents a factor not discussed thus far and generally not applicable to technical augmentations: depth. In the context of a camera taking an image, non-technical augmentations can typically occur in one of four places: On the lens, on a protection window in front of the lens, between the lens and the object, or on the object. This is a significant consideration for water-based effects which can take a different form depending on where it occurs – for example, water could appear as rain when it occurs between the camera and its target or droplets when it occurs on the camera. For this thesis, it is assumed that outdoor cameras are placed behind an intermediary protective window, preventing augmentations occurring directly on the lens.

Most rain-related papers depict rain as it occurs between the camera and object rather than on a window in front of the lens [81-83, 87]. In fact, generation of raindrops on an intermediary window occurs more commonly in the context of video game effect creation than within artificial augmentation generators. For this reason alone, raindrops make for an interesting candidate for generation. Furthermore, this presents an interesting dilemma when considering a cameras' autofocus function. If the camera focuses on the raindrops, it is likely to produce a vastly different augmentation than if it were to focus on its intended object.

Based on the investigations described within Section 2.3, nine contextually relevant augmentations have been proposed as candidates within Table 2.6 for artificial generation for this project. It is believed the nine selected augmentations represent a healthy range of

explorable effects achievable within the timeframe of this project. This list is comprised of two, easily replicable augmentations and six, more advanced augmentations which are much less readily found in online datasets / generators.

Table 2.6 Proposed list of generatable augmentations

Augmentation	**Desired emulation of…**
Brightness	General environmental brightness
Darkness	General environmental darkness
Motion Blur	Motion due to wind or moving sensor
Sun Flare	Localised increased brightness due to sun
Dirt	Wet mud, fine dust on protective window
Focused Raindrops	Raindrops on protective window and in focus
Unfocused Raindrops	Raindrops on protective window and out of focus
Frost	Ice / Condensation on protective window
Water Residue	Residue left from dried up raindrops on protective window

Each of the nine effects listed in Table 2.6 are effects that an edge device could be expected to encounter in an uncontrolled outdoor environment. Within this environment, it is often difficult to find consistency within produced augmentations since environmental conditions are a result of a number of factors such as the time of day, season, and geolocation. Accounting for all of these variances is a difficult task and as a result, augmentation generators tend to either ignore or hardcode certain factors in order to make generation more manageable. In order to make a useful generic augmentation generator, end users require the ability to configure generic augmentations to fit their individual needs. This necessitates a level of configurability.

2.4. Summary

From the perspective of both image recognition and object detection performance, this chapter has explored the basis of deep learning models, their vulnerability to augmentation and the current approaches to mitigating this vulnerability making a deep learning model robust. Deep learning models are prone to overfitting and hence can be difficult for a model to generalise. As this is one of the many core features of human vision, it is an important prerequisite for true replication of computer vision techniques such as IR or OD. Model robustness is a measure of how well a model performs given augmented data when compared to clean, unaugmented data. This chapter has discussed four approaches that can be used for robustness improvements, each with their own situational advantages and disadvantages. Robustness improvements are a vital improvement in making current models more accessible in safety-critical industries such as forestry, healthcare or aviation by improving expected model reliability.

The scope used in this thesis prevents the usage of all robustness methods discussed in this chapter with the exception of data augmentation, making it the best robustness approach for this project. Furthermore, it can be argued that data augmentation is the most widely applicable approach of those discussed as it is the only approach not requiring either additional processing resources or a specific understanding of the weak points of a model. These dependencies can make implementing robustness improvements undesirable, especially for model implementers who are working with limited resources or do not have the time or ability to determine weak points within the model they are trying to implement. Section 2.3 explored existing augmented datasets and augmentation generators with the aim of identifying how deeply this topic has been explored. This exploration identified that there exist many augmented datasets, many augmentation generators but nothing truly universal aimed at the mass generation of diverse, contextually relevant augmented datasets.

Table 2.6 summarises nine augmentations to be used as proof-of-concept augmentations for this project. These represent a variety of augmentations an edge device would likely encounter within an outdoor environment. Artificial generators for each of these nine augmentations will be created for mass-application on an unaugmented dataset as fulfilment of their proof-of-concept status. In the following chapter, the prerequisites for the generation of these augmentations are explored, followed by a chapter discussing the methodology behind the generation of the nine selected augmentations.

3. Prerequisite Tools and Techniques

This chapter introduces the benchmarking datasets and evaluation techniques that will be used throughout the remainder of this thesis. These datasets will be used to explore the validity of using the nine selected artificially generated augmentations on existing real-world datasets to improve the robustness of models already trained on these datasets. The datasets chosen for this evaluation are important as they will have a direct influence on the evaluated performance. In the context of this work, poor quality datasets are considered to be those that already contain augmentations, contain diversity within their collective image features, or are otherwise inconsistent. Pre-existing augmentations will likely have an unknown effect on model performance, invalidating the certainty of any results produced making pre-existing augmentations undesirable for this thesis.

Three datasets were selected for evaluation, and these are discussed within this chapter. For each dataset, a baseline metric is determined which is used in Chapter 5 to evaluate the relative performance of the nine chosen augmentations. Each of these datasets were specifically selected to fit within the scope of this project, meaning that the captured images were taken from within an uncontrolled, outdoor environment where they can be realistically subjected to the nine augmentations described in Chapter 4. Both scope compliance and the expectation that the nine chosen augmentations could occur realistically within the chosen datasets introduces restrictive criterions, but is crucial for ensuring meaningful results. Another crucial criterion is that each dataset must be labelled, as labels are required for evaluation. It is important that all datasets meet the criteria listed above in order to achieve consistent and fair evaluations. For clarity, these criteria will be restated:

- Does not contain pre-existing augmentations.
- Is contextually relevant to project scope.
- Can realistically be subjected to any of the nine effects listed in Table 2.6.
- Must be labelled for usage in supervised learning.

Of the three chosen datasets, two are used for image recognition evaluation and one for object detection. These datasets were used to explore the impact of applying artificially generated augmentations at five increasing intensities and the results compared to an established baseline. In addition, real-world case studies which introduce another two datasets are used to determine

the effectiveness of using artificial augmentations instead of natural augmentations. These experiments will be discussed in Chapter 5.

The ‘DeepWeeds’ and ‘Wellington Camera Traps’ datasets both stem from published research papers and are labelled for image recognition. The third dataset was created for this project using images sourced from the larger Open Images V6 dataset. Each of these datasets have slightly different size, perspective, class, diversity, and usage specifications which is intended to expand the variety of image types in which to apply artificial augmentations. However, despite this, all three datasets remain compliant with the above established criteria. These datasets are introduced in Sections 3.3, 3.4 and 3.5.

As each dataset has been created by different people, all three have been prepared differently. In order to achieve fair comparison, each dataset is prepared specifically for this project such that a measurable baseline model can be established. This helps ensure that all image recognition datasets are trained and evaluated in the same manner such that no performance differences can be attributed to anomalies presented by usage of different image recognition frameworks. Failing to ensure consistency in data preparation can result the creation of invalid, and ultimately useless data.

For image recognition, this thesis uses a generalised version of the DeepWeeds training program which is introduced in Section 3.2. For object detection, ‘You Only Look Once’ (YOLO) v5 will be used. YOLO v5 is a fairly new object detection framework and although it is not included in list of object detection frameworks in Table 2.3, it is an improvement on previous versions of YOLO that can be found in this table and is well known for its ability to achieve similar mAP scores in less time than comparable object detection frameworks.

To further ensure training quality, cross-validation was used. Cross-validation is a technique used to try find the most robust model produced through utilisation of different training/testing splits of a dataset. This technique is typically used when training on either a small or imbalanced dataset. In such cases, a dataset may not have enough images to fully teach a model to recognise a particular class or a dataset may not have enough variety within images reinforcing generalisation issues. Cross-validation is most often used to ascertain the best training set for creation of a model that performs well on unseen data and can be performed in a variety of ways, Section 3.1 introduces how it was applied in this thesis.

To further ensure training quality, 5-fold, leave-one out cross-validation was used for model training. This technique is discussed in Section 3.1. This chapter is concluded with a summary in Section 3.6 which recaps evaluated baseline models in preparation for Chapter 5.

3.1. Cross-Validation

In a normal training approach, datasets are prepared for training by splitting them into three subsets: training, validation, and testing. The training subset normally requires proportionally more images than either testing or validation subsets to ensure a model has sufficient images to produce a viable model. For example, a dataset might be split into subsets using a 70:15:15 ratio, where 70% of the images are allocated for training, 15% for validation and the remaining 15% for testing. While it is possible to maintain a fair class balance when splitting a dataset into these subsets, this can be difficult to achieve. This means that there is a possibility for any class that one subset will contain image features that are not found within other subsets, resulting in a model underperforming when encountering either unseen new features, or missing features in an image. Cross-validation mitigates this issue by training and evaluating multiple copies of a model, each with a different distribution of images over the three subsets. It is worth noting that despite sharing similar names, cross-validation and the validation subset are not related. The validation subset is used during model training to evaluate the model's performance and dynamically adjust parameters within the training progress, effectively validating that the model is learning optimally.

When performing traditional cross-validation, the validation subset is omitted, and k unique testing subsets, called 'folds', are created, effectively creating k unique distributions of a dataset. An important rule of fold creation is that each image within the whole dataset can only occur in a fold once. As a result, it is assumed that the remaining images not in the fold are to be used for training.

In other methods such as nested cross validation and leave-one-out cross validation, the validation subset is included by creating a validation fold which, as with the test fold, each image can only occur in a validation fold once. In these methods, any image not part of either the validation or test fold is available for training.

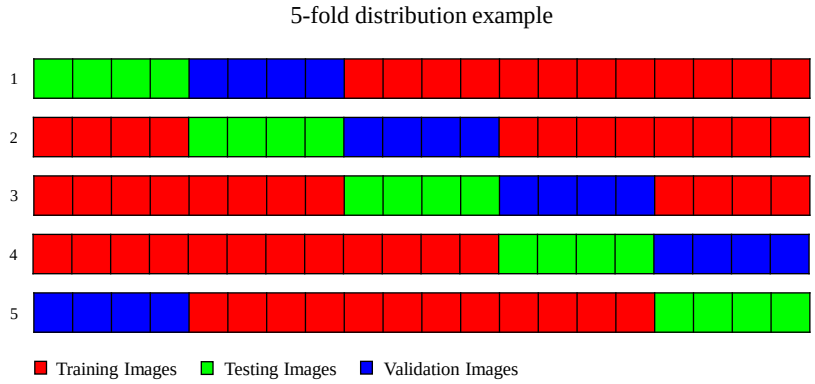


Figure 3.1 Leave-one-out 5-fold cross-validation distribution

This thesis uses 5-fold, leave-one-out cross validation for all model training. Leave-one-out cross validation is the simplest method of creating unique validation folds. Figure 3.1 shows five rows, each depicting a twenty-image dataset that has been split into three subsets whilst keeping the validation and testing images unique across each. The result of this process is five models which have been trained on different variations of the same data. When evaluated on its remaining test data, it is expected that each model will perform slightly differently. For production purposes, this technique can be used to identify the model that performs the best on its unseen test images under the assumption it is a stronger model. In this paper however, no models will be discarded. It is expected this technique will help identify anomalous errors in a particular trained model and ensure results are not skewed as a result of either an overperforming or underperforming model.

3.2. Image Recognition Training and Evaluation Program

As part of a fair experimentation, both the DeepWeeds dataset and the Wellington Camera Traps dataset must be trained and evaluated using the same software. Using the same program ensures that any anomalies that occur as a result of using a particular image recognition framework are applied consistently to all models such that in theory, no model has an advantage over another. It is important to ensure that with the exception of the dataset used, the models are identically trained.

The authors of the DeepWeeds dataset have provided a training and evaluation program that in Section 3.3 is shown to perform very well. There was no training program provided for the Wellington Camera Traps. Since the DeepWeeds training program is already setup for cross-validation, the DeepWeeds training program was used for all image recognition tasks in this thesis.

The DeepWeeds training program is written in Python and can be found on GitHub [1]. The program utilises the TensorFlow framework with Keras [88], which is a high-level API that provides a more user-friendly interface for interacting with TensorFlow. The creators of this program also added a number of additional features such as a variable learning rate which can be lowered once a plateau in learning has been detected. Lowering the learning rate means that training makes smaller adjustments to the model based on the models' predictions rather than bigger ones, effectively fine tuning the models' weights. Another feature is image randomisation which helps reduce overfitting by applying lightweight, simplistic transformations such as rotation and contrast adjustments. This is a common feature of many images' recognition tools and has already been discussed in Section 2.2, but technically does apply very minor augmentations to the clean dataset. However, as these minor augmentations are commonly applied by most image recognition programs and will also be applied to the augmented variants of the datasets. They were not removed as they are unlikely to be removed in non-experimental research.

The DeepWeeds training program was designed to utilise either a ResNet-50 or InceptionV3 model which has been pre-trained on the ImageNet dataset. As the ImageNet dataset images are 224 by 224 pixels in size, the program resizes input images by cropping the image down to the middle 224 pixels in width and height. This presents an issue when using this training program on datasets with larger images, such as in the case of the Wellington Camera Trap dataset. Cropping images removes data from the image, potentially rendering labels untrue. Downscaling images to 224 by 224 was a better, and more common approach to this problem.

The authors of the training program supplied a requirements file specifying the packages required to utilise the program. Unfortunately, the file only lists minimum package versions which results in the latest version of the listed packages being sourced. Despite installing the

[1] https://github.com/AlexOlsen/DeepWeeds, accessed 23/03/2023

listed minimums, a number of errors were still encountered that prevented running the program. These issues, and the steps taken to resolve them, are listed in Appendix A.

The majority of changes made to this program, with the exception of those listed in Appendix A, were introduced to generalise the usage of the program. Largely, this meant replacing hardcoded variables such as 'IMAGE_DIRECTORY' with a runtime argument. Additionally, a few usability changes were implemented to assist with the automation required to sequentially evaluate multiple datasets without requiring manual intervention. For convenience, the cross-validation and evaluation functions were separated into two distinct programs which offer task-specific command line options. The only significant change made to either programs' process-flow is the replacement of the crop operation with a resize operation to mitigate the issue described previously within the cross-validation program. A copy of the modified programs are available on request to the author.

3.3. Dataset: DeepWeeds

The DeepWeeds dataset contains a collection of eight species of Australian weeds taken from a top-down perspective [89]. In total, there are 17,509 images split over nine classes, where the ninth class is a catch all class for images not containing one of the eight classified weed species.

This dataset not only provides labels for supervised learning, but also provides five pre-created subsets for cross validation of the dataset. As such, it is a suitable candidate for this project.

Table 3.1 Distribution of classes across locations showing distribution of classes overall images [89] (Apache License 2.0)

		Locations								
		Black River	*Charters Towers*	*Cluden*	*Douglas*	*Hervey Range*	*Kelso*	*McKinlay*	*Paluma*	Total
Weeds	*Chinee apple*	0	0	0	718	340	20	0	47	1125
	Lantana	0	0	0	9	0	0	0	1055	1064
	Parkinsonia	0	0	1031	0	0	0	0	0	1031
	Parthenium	0	246	0	0	0	776	0	0	1022
	Prickly acacia	0	0	132	1	0	0	929	0	1062
	Rubber vine	0	188	1	815	0	5	0	0	1009
	Siam weed	1072	0	0	0	0	0	0	2	1074
	Snake weed	10	0	0	928	1	34	0	43	1016
	Negatives	1200	605	1234	2606	471	893	943	1154	9106
	Total	2282	1039	2398	5077	812	1728	1872	2301	17509

In Olsen, et al. [89], this dataset achieves an impressive 95.1% and 95.7% on the Inception-V3 and ResNet-50 ImageNet pretrained models respectively. These percentages reflect the weighted average of the Top-1 accuracy metric. Using the Top-1 metric instead of the F1-Score metric on an imbalanced dataset can be misleading when evaluating unevenly distributed datasets, as poor performance in a lesser represented class can be overshadowed by excellent performance in a better represented class. However, as will be discussed in Section 3.3.2, the Top-1 averages reported by Olsen, et al. [89] do indicate a consistent performance across all classes by both models, resulting in average F1-Scores for the fivefold sitting above 90%.

While all images in this dataset have been captured in a controlled manner by ensuring the consistency of similar weather conditions and camera placement, the environment in which these species exist is naturally volatile and otherwise difficult to control. As the quality of these images are so consistently high, this dataset serves as a suitable candidate for the addition of artificial augmentations providing the convenience of images not having pre-existing augmentations that could interfere with results. Furthermore, as this dataset has been shown to perform well when trained on a pre-trained Inception or ResNet model, this allows for a higher range of performance impacts which in theory should better highlight the impact that different augmentations can make on model performance.

Finally, as this dataset is labelled, contains no pre-existing augmentations, and is comprised of images that could be realistically expected to be subjected to any of the nine listed augmentations, this dataset satisfies the criteria introduced in Chapter 3.

3.3.1. Environment Preparation

All images of the DeepWeeds dataset are 256 by 256 pixels, JPEG images. Two sets of labels are provided alongside the images as CSV files. The first set is a single CSV file which contains the filename, class id and species name for each image. The second set, designed for 5-fold cross-validation, is a collection of 15 csv files where the dataset is split into five folds, each distributing images using a 60:20:20 ratio for training, validation, and testing respectively. No additional processing is required to prepare this dataset for usage.

3.3.2. Establishing Baseline Model

In Olsen, et al. [89], the DeepWeeds dataset performed very well achieving 95.1% and 95.7% on the Inception-V3 and ResNet-50 models respectively. These results represent the average Top-1 accuracy score across the results of a 5-fold cross-validation. Each fold took approximately 13 hours of training on an Nvidia GTX 1080Ti GPU. For this thesis, both the Inception and ResNet models were re-trained using the default configuration provided by the original DeepWeeds training program.

Table 3.2 Training + Validation performance metrics of ResNet-50 model on DeepWeeds dataset

Model	**Metric**	**Project Average**	**Olsen, et al.'s Average**
ResNet	**Top-1%**	94.9%	95.7%
	F1-Score	93.4%	--
Inception	**Top-1%**	92.8%	95.1%
	F1-Score	90.9%	--

Table 3.2 provides a comparison between the average Top-1 performance achieved for each model by both this thesis and by Olsen, et al. [89]. Despite using the same folds and initial pre-trained weights as used by [89], the Top-1 averages achieved by this thesis are both lower indicating comparatively worse performance. It is difficult to argue as to what level of difference between each model's reported averages are acceptable. This is made harder as both randomness and the required adjustments to the training program as listed in Appendix A have

an unquantifiable impact. Hence, it is much easier to accept the 0.7 percentage point difference between the two ResNet averages as being negligible than it is the 2.3 percentage point difference between the Inception averages.

In addition to the Top-1 metric, the F1-Score has been reported for each model trained for this thesis. This metric cannot be compared to, or calculated from [89], however provides a comparison for the other F1-Scores discussed in this thesis. The confusion matrix presented in Table 3.3 validates the performance consistency for the ResNet model over each of the nine classes indicating that the Top-1 percentages reported in Table 3.2 are not misleading. This is evident through the comparison of high percentage true positive values shaded in green to the non-shaded, low percentage false positives and negatives.

Table 3.3 Summed confusion matrix for ResNet model as percentage. Green shaded cells represent true positive values, horizontal columns represent false positives, vertical columns represent false negatives

		True Class								
		Chinee Apple	**Lantana**	**Parkin-sonia**	**Parthen-ium**	**Prickly Acacia**	**Rubber Vine**	**Siam Weed**	**Snake Weed**	**Negatives**
Predicted Class	**Chinee Apple**	84.1	1.9	0.0	0.5	0.4	0.2	0.3	5.1	7.6
	Lantana	0.7	95.2	0.0	0.0	0.0	0.1	0.5	0.9	2.5
	Parkinsonia	0.1	0.0	97.2	0.3	0.8	0.1	0.0	0.0	1.6
	Parthenium	0.4	0.2	0.5	94.6	1.7	0.3	0.1	0.2	2.1
	Prickly Acacia	0.2	0.0	1.2	1.1	93.9	0.0	0.0	0.1	3.5
	Rubber Vine	0.4	0.5	0.0	0.1	0.0	93.2	0.2	0.6	5.1
	Siam Weed	0.0	0.3	0.0	0.0	0.0	0.0	96.4	0.0	3.4
	Snake Weed	3.3	2.0	0.0	0.5	0.0	0.2	0.4	89.4	4.2
	Negatives	0.5	0.4	0.3	0.2	0.7	0.1	0.5	0.5	96.7

As a result of both the Inception and ResNet models achieving similar F1-Scores, continued evaluation of both models was dropped in favour of the sole evaluation of the slightly better performing ResNet model. This in part was due to time constraints, but also under the assumption that the time invested into evaluating augmentations on both models would yield anything more significant than evaluating augmentations on the ResNet model alone.

Table 3.4 Baseline average for DeepWeeds dataset

Metric	Baseline Average
F1-Score	95.3%

Table 3.4 reports the baseline average that will be used for the evaluation of augmentations on the DeepWeeds dataset as 95.3% when evaluated on a DeepWeeds trained, ResNet pre-trained model.

3.4. Dataset: Wellington Camera Traps

The Wellington camera traps dataset contains 270,450 images sourced from 187 cameras within Wellington, New Zealand [90]. Each camera was setup to capture three burst images over a two second period when triggered by pest movement nearby. These images have been classified into seventeen classes which includes fifteen animal classes, an empty class, and an unclassifiable class.

The images were collected using remote cameras and uploaded to a Victoria university of Wellington public animal identification website [2] for labelling by interested members of the public. Labellers were tasked with classifying each image sequence (collection of three burst images) with one of the categories along with their confidence. All sequences were classified by at least three members of the public and two formally qualified experts to attempt to ensure the quality of the assigned classifications.

Many of the images in this dataset were taken at night under infrared lighting, as a result, this dataset presents a large range in the quality of its images with some being full colour, well-lit images, and others grayscale. It was expected that this would make analysis of the impact of certain augmentations more difficult, especially with the utilisation of infrared lighting, through added complexity as arguably, the usage of infrared lighting to make a greyscale image could be considered an augmentation in itself. Additionally, each image has had a watermark applied.

In its published state, this dataset does not conform with the criteria defined in Chapter 3 as it contains pre-existing augmentations. However, this dataset contains a number of attractive

[2] http://www.identifyanimals.co.nz/, accessed 29/03/2023

features which in comparison to the DeepWeeds dataset make this dataset desirable, such as a larger image size and ratio between object and background sizes. These features allow for a wider analysis of the application of artificially generated augmentations while still producing results that can be compared to the DeepWeeds dataset. In order to make this dataset comply with this chapter's criteria, images with pre-existing augmentations must be removed at a minimum.

The original dataset can be obtained from LILA BC [91].

3.4.1. Environment Preparation

Although this dataset is ready for usage in image recognition tasks, this dataset was originally used to investigate the effectiveness of using inclined volunteer citizens to facilitate image classification. Image classification on this dataset is investigated in another paper by Shashidhara, et al. [92], which explores improvement of image recognition through utilisation of burst sequences rather than a single image. However, in [92] the authors chose to perform a binary classification of either 'animal' or 'no animal' rather classify all seventeen categories. This is achieved by re-classifying all empty and unclassifiable images as 'no animal' and the remaining fifteen categories as 'animal'.

Size reduction. In its zipped form, the Wellington Camera Traps dataset is 176GB, and naturally, even larger when decompressed. As nine copies of this dataset, each with one of the augmentations listed in Table 2.6, would be generated, an excess of 1.7TB would be required. Even before considering the storage requirements for the other datasets, this volume of data exceeds the space available for utilisation by this project. Furthermore, working with a dataset of this size in a distributed manner, such as through the utilisation of either external storage mediums or network file shares would likely result in lengthy file transfer times as a result of copying numerous small files ultimately hindering the completion of this project within the allocated time. The large size of the dataset stems from a combination of the size and total number of images within the dataset, with each image measuring 3264 by 2448 pixels.

To decrease the overall dataset size, two actions were taken:

Firstly, two thirds of the dataset was removed by deleting additional burst images. As each image within this dataset is a member of a three-image burst sequence, where each image

within the sequence was taken 0.6 seconds after the previous, it can be assumed that the difference between images within a sequence is minor and that the second and third image in each sequence is a redundant copy of the first image. Removal of these images results in a large decrease in the overall dataset size, and as labels are applied at an image level rather than a sequence level, removal of these images does not invalidate the remaining images' label.

Secondly, each remaining image is downsized by 75% to further reduce the size of the overall dataset. This is a significant decrease in each image's overall size, however, becomes slightly less significant when trained using the program introduced in Section 3.2 as this program downsizes all input media to 224 by 224 pixels. Furthermore, it is unlikely that an edge device would be expected to either perform image recognition or transfer images of this size due to the demand on the device's resources. Although this decrease will result in a large loss of data, the same downsize was performed by Shashidhara, et al. [92] where they achieved a 0.852 ROC AUC score on a ResNet-50 model. 'Receiver Operating Characteristic Area Under the Curve' (ROC AUC) is reported as a number between 0 and 1 and is commonly used on balanced binary datasets. This metric reflects how well a model can distinguish between two classes (positive and negative) by plotting each true positive rate versus false positive rate. Simply put, for this scenario, a high ROC AUC metric indicates a model is good at correctly classifying images as either 'animal' or 'no animal'.

Reclassification. Following the classification process used in [92], all images have been relabelled as either 'animal' or 'no animal'. Reclassifying each image removes the need for models trained on this dataset to need to identify specific animal types, simplifying the overall detection process which should reduce the number of incorrect predictions amongst similar looking animal species.

Figure 3.2 Image from Wellington Camera Trap showing in red the data lost if water mark removed. Wellington Camera Trap images adapted from [91] (Dataset, https://cdla.dev/permissive-1-0/)

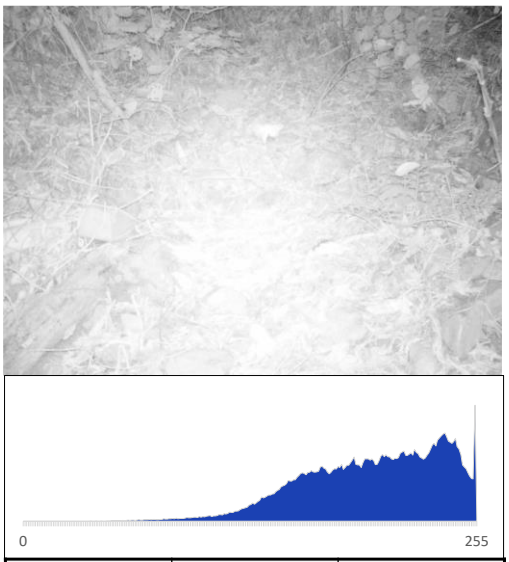
Watermark Removal. Each image in the dataset contains a watermark running along the bottom most border of the image as can be seen in Figure 3.2. Although it is consistent among the dataset, this watermark has the potential to interfere with model training if the model decides to learn features of the watermark. Furthermore, leaving the watermark in introduces a significant number of white pixels which could interfere with the detection of pre-existing augmentations, discussed next.

All watermarks have been removed from this dataset by cropping out the pixels shown within the red bounding box in Figure 3.2. Removing the watermark also removes an arguably insignificant number of non-watermark pixels, however, as can be seen in Figure 3.2, does still run the risk of removing pixels that represent an animal. This means that by removing the watermark, there is a potential to invalidate an images label, although, it is expected that Figure 3.2 represents a worst-case scenario and which is not reflective of the majority of the images within the dataset.

Pre-existing augmentation removal. Three kinds of pre-existing augmentations were found in this dataset making images: overly dark, overly bright or greyscale. All images containing one of these augmentations were removed to ensure compliance with the criteria specified in Chapter 3.

Greyscale images can be detected by tallying the count of each byte value used for each channel and measuring the distance between each channel's most common byte value. If an image is mostly greyscale, the most common byte for each of the channels, or channel peak, will occur close together.

Max R	Max G	Max B
47	255	255
$\|R - G\|$	$\|G - B\|$	$\|B - R\|$
208	0	208



Max R	Max G	Max B
254	254	254
$\|R - G\|$	$\|G - B\|$	$\|B - R\|$
0	0	0

Figure 3.3 shows the RGB component distribution across each image's pixels. Greyscale images can be identified by images that have a small distance between the most common byte of each channel. Wellington Camera Trap images adapted from [91] (Dataset, https://cdla.dev/permissive-1-0/)

Figure 3.3 shows a RGB histogram which represents the occurrence of each byte value within each of the three channels over every pixel in the image. Both images have had their watermarks removed for this analysis as the watermark seen in Figure 3.2 adds a significant number of high valued pixels that could result in an image being incorrectly filtered. The left-hand side image is not greyscale which is reflected in the histogram by showing a significant gap between at least one of the peaks. The histogram under the right-hand side image however does not show any well-spaced peaks indicating that the most common byte of each channel are either the same, or very close neighbours meaning most pixels are a shade of grey. Under each histogram are two stacked tables. The topmost tables list the most common byte as shown in its respective histogram for each channel. The second table calculates the distance, or the

number of bytes, in between each channel most popular byte. As the calculated distances can range anywhere from 0 through 255, a threshold value is used to determine how close two channel peaks must be in order to be considered greyscale such that an image is considered to be greyscale if all three distances are below the threshold. This thesis found that a threshold of 10 worked sufficiently for this process, hence, the right-hand image is considered greyscale and the left-hand is not. This approach can cause issues when an image is intentionally greyscale, however, this is not the case for any image within this dataset.

Overly dark or bright images are be removed by requiring the average of the most common byte values for each channel to fall within a desired range. For this thesis, a number of values were trialled, but a range of 25 – 235 was found to result in an acceptable collection of images which were neither overly bright nor dark. The left-hand image in Figure 3.3 has an average peak value of 186, suggesting it is neither too bright nor dark whereas the right-hand image has an average peak value of 254 suggesting it is too bright.

Cross-validation splits. The final change was to redistribute the images into five folds for 5-fold cross-validation. This was achieved through a script which performed the same sliding window approach as shown in Figure 3.1. This resulted in the same fifteen CSV label files provided with the DeepWeeds dataset.

After preparation, this dataset was reduced to 11,304 images with 8,920 labelled as ‘Animal’.

3.4.2. Establishing Baseline Model

The performance metric for the Wellington Camera Traps dataset calculated in this section represents the baseline performance of the models trained in this section on unaugmented data. As discussed in Section 3.4.1, the dataset has been significantly reduced and altered making a comparison between values reported in [92] unfair. For this project, the Wellington Camera Trap models were trained under the same conditions as reported in Section 3.3.2 that were used to train the DeepWeeds models. On average, each model took 53 hours to train.

Table 3.5 Training + Validation performance metrics of Wellington Camera Traps model

Metric	**Training + Validation Average**
Top-1%	80.4%
F1-Score	60.6%

Table 3.6 Summed confusion matrix for Wellington Camera Trap cross-trained models as a percentage. Green shaded cells represent true positive values, horizontal columns represent false positives, vertical columns represent false negatives

		True Class		
	CLASS	Animal	No Animal	Total
Predicted Class	Animal	95.8	4.2	8917
	No Animal	77.2	22.8	2383

Table 3.5 reports both the average Top-1% and the F1-Score metrics achieved across all five folds after training. Despite achieving a Top-1% of 80%, the class distributions shown by the green-shaded cells in Table 3.6 show that the well performing Animal class with a significantly higher number of images is artificially inflating the accuracy of the model. This scenario reinforces the importance of the F1-Score metric which punishes the model's poor performance in the No Animal class resulting in an F1-Score of 61%.

Table 3.6 shows that over all trained models, there is a bias towards the Animal class. There are many possible reasons for this such as there being an insufficient amount of training data remaining after culling, too much data loss after downsizing, the Animal class incorrectly being treated as a negatives class for low confidence detections, or that the models have genuinely struggled to learn to differentiate the classes. Despite these values, it is not the aim of this project to train well-performing models, rather to evaluate the impact of augmentation in a realistic environment. The overall F1-Score of 60% is of sufficiently high enough value to see performance adjustments as a result of the presence of augmentation.

Table 3.7 Baseline average for the Wellington Camera Traps dataset

Metric	**Baseline Average**
F1-Score	60.5%

Table 3.7 reports the baseline average derived from the test data that will be used for the evaluation of augmentations on the Wellington Camera Traps dataset as 60.5%.

3.5. Dataset: Open Images V6 Trees

Open Images is a large dataset containing millions of labelled images, which is continually being added to. It was first released by Google in 2017 with Open Images V6 being the latest version when experimentation began [93]. Open-Images provides labels for a variety of computer vision tasks including image recognition and object detection.

This dataset was primarily created for an evaluation of artificially generated augmentations on a real-world object-detection case study that is introduced in Section.5.3. Evaluation on these real-world scenarios is an important step in evaluating the usage of artificially generated augmentations in the place of naturally occurring augmentations for robustness training. Furthermore, it shows that artificially generated robustness improvements are not restricted to image recognition tasks.

Both studies were performed within the context of an outdoor, urban environment and share trees as a common object class in their respective images. Open Images V6 enables the creation of a tree dataset for model training by enabling the selective download of images containing the tree classification. The download can be achieved using a tool called FiftyOne [94] which enables fine-grained selection using python [95]. The FiftyOne command used to create the tree dataset is listed under Appendix C.

After the removal of empty images, the downloaded dataset consisted of 14,000 training images, 3,000 validation images and 3,000 testing images. A broad range of trees can be seen within the downloaded images, some of which also contain pre-existing augmentations. Unlike with the other datasets, this is acceptable as this dataset is intended for creation of a model which is used in Section 5.3 on a real-world case producing a dataset that conforms with the criteria specified in Chapter 3.

The unaugmented dataset used in this thesis is available alongside the generic DeepWeeds training program on request to the author.

3.5.1. Environmental Preparation

Two changes were required to the tree's dataset. The first involved reformatting the images' labels into a format accepted by YOLO v5, and the second involved creation of cross-validation splits.

Label reformatting. YOLO v5 expects a text file containing the location of an object on each row to accompany each image. The labels provided by Open Images V6 do not meet this standard. This issue is fairly easy to resolve, however, does require recalculation of the bounding boxes to satisfy YOLO's labelling system which introduces the possibility for error.

Cross-validation splits. Creation of 5-fold cross-validation splits was achieved by disregarding the pre-prepared subsets in the downloaded dataset and using the same sliding window approach used to create the folds for the Wellington Camera Traps dataset. One significant difference for this dataset was that rather than create only new label files, five copies of the whole dataset were created and organised into their correct subsets to simplify interaction with YOLO.

The YOLO V5 processor was obtained from GitHub and did not require any adjustments to operate [3].

3.5.2. Establishing Baseline Model

YOLO provides a collection of pre-trained models of various model sizes as a starting point for model training. All pre-trained models were trained to 300 epochs using default settings. As with most deep learning models, larger models tend to have better performance accuracy however take longer to train. Ultimately, selection of a pre-trained model is dependent on the amount of data and features that can be extracted from it. To establish the baseline model, the large pre-trained variant, "YOLOv5L" was used as after some testing, appeared to offer a better performance over smaller pre-trained models with a comfortable training time.

[3] https://github.com/ultralytics/yolov5/

Each cross-fold was run individually, but on the same hardware and using the same training program. The mAP and mAP@[0.5:0.05:.95] scores for each fold can be found in Table 3.8.

Table 3.8 Testing mAP metric scores for each tree-fold

Metric	**Fold 1**	**Fold 2**	**Fold 3**	**Fold 4**	**Fold 5**
mAP	55.1%	59.4%	37.2%	30.7%	49.5%
mAP@[0.5:0.05:.95]	30.7%	33.1%	17.9%	12.5%	25.7%

There is a significant range in the metrics produced by each fold, likely indicating that folds 3 and 4 were trained on data that was either labelled inconsistently to other data samples or contained significantly different features to that of the images used to calculate mAP metrics. Figure 3.4 contains two images from the dataset which both contain distinct tree trunks but are labelled differently. This presents a rather unique problem to tree detection in that when trees are clustered, they can easily appear as a single tree even though they are technically separate entities. The inconsistent labelling is a result of this problem, as the dataset has likely been labelled by a large collection of people, all having a different perspective on when a single tree becomes a tree group. In addition to this is the challenge of creating a bounding box that best represents a tree without the inclusion of either background tree-like image features. This is another labelling decision that differs as a result of the perspective of labeller.

Figure 3.4 Comparison of 'Tree Group' versus 'Tree' labelling

Naturally, the trained models will try to fit best with their provided labels such that the predictions made by a model should amalgamate the various perspectives of the truth labellers.

This has an impact on the mAP metric whereby if the predicted bounding box does not match the perspective of the particular labeller that labelled the image being evaluated, the prediction is classified as a false negative. This effect is demonstrated in Figure 3.5, which provides a comparison of the truth versus predicted labels for an image in the dataset. The true labels for this image identity three tree groups whereas the tree model identifies a single tree group in the same area as well as a handful of trees. Focusing on only the tree group detection, depending on the individual perspective, the model's prediction is arguably more correct than the true labels, however, it is an incorrect prediction in the context of mAP.

Figure 3.5 Comparison image between true label and predicted label for a subjectively poorly label image

Based on this discussion, it is hard to definitively state that the mAP values listed in Table 3.8 indicate poor performance. With sufficient time, the dataset images could be relabelled by either a single person or a small group of people with a classification criteria to determine the impact of the inconsistent labelling on the resultant mAP scores. However, regardless of the specific value of the metric, this averaged mAP metrics listed in Table 3.9 which are computed from the values in Table 3.8 will allow for an adequate comparison to augmented versions of the tree dataset.

Table 3.9 Established baseline mAP scores for Trees model

Metric	**Baseline Average**
mAP	46.4%
mAP@[0.5:0.05:.95]	24.0%

3.6. Summary

This chapter has introduced the DeepWeeds, Wellington Camera Traps and the Open Images V6 Trees datasets that will be used for evaluation of augmentations in Chapter 5. The DeepWeeds and Wellington Camera Trap datasets are image recognition datasets, trained and evaluated using the generic DeepWeeds training program introduced in Section 3.2. The Open Images V6 Trees dataset is on object detection dataset which this program trains and evaluates using YOLOv5.

Each dataset has been introduced, prepared for evaluation be ensuring consistency between images and creating cross-validation labels where needed such that cross-validation training can be used in this chapter to establish baseline performance metrics for each dataset.

4. Methodology

This chapter begins by introducing the artificial augmentation generator program used to render each augmentation onto an unaugmented dataset to create an augmented dataset. The chapter then continues to describe, in detail, the generation process of each augmentation before introducing the augmentation pre-sets and describing how each dataset was prepared.

4.1. Artificial Augmentation Generation Program

Rather than create a separate program or script to generate each of the augmentations, a modular augmentation program was created to facilitate both the creation and bulk application of augmentations to image datasets. This program is designed to be run interactively when previewing and configuring augmentation properties, and through the terminal when performing bulk augmentation applications.

Each of the augmentations listed in the following sub-sections contains a collection of configurable properties which help to influence various aspects of the augmentation. When run interactively, the program allows users to experiment with these properties and create pre-sets for augmentations. Pre-sets are used for bulk augmentation and provide either fixed or variable values for augmentation properties. To ensure variation between augmentations, settings can be used to specify a maximum and minimum range for random value selection as well as to perform minor adjustments on subsequent generations of the same augmentation. This is achieved by also providing a variance value when creating a variable pre-set property, and helps to provide incremental change between augmentations when desired. Figure 4.1 demonstrates an example variable property used to select a random compass bearing. The properties minimum and maximum are defined as 90° and 210° respectively are indicated by the green line. If a random value of 120° is selected from this range, the next random number must occur within ±15°, indicated by the blue line. The variance value does not permit selection of a random number outside of the minimum and maximum range.

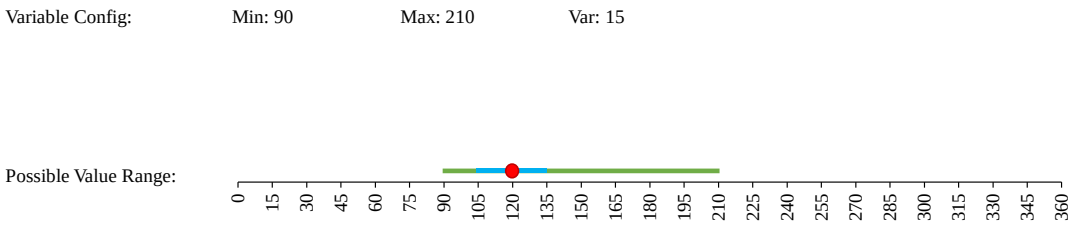


Figure 4.1 Implementation of variable pre-set properties

All nine of the augmentations listed in the following sub-sections have been created using Open Computer Vision (OpenCV), which is a widely used computer vision library written in C++. As this augmentation generation program is written in C#, the OpenCV Sharp library was used as an interface. All augmentations generated for Section 4.2 were generated on an Ubuntu 18.04.6 LTS server using the MONO runtime framework, as result, there is no dependence on either Windows or the .Net framework for this program's operation.

This program was designed to be extended without modifying the program through user plugins allowing the program to apply augmentations other than the nine created below. More information on this can be found in the project's repository linked in Section 1.2.

When working with augmentation code, images are typically represented by one or more matrices depending on the number of colour channels contained within an image. This means that for most images, either three or four matrices are required to represent an image depending on whether the alpha channel is present. Unless stated, it can be assumed in this section that each equation is applied individually, but identically, to each of these matrices and hence for simplicity, equations will represent input and output images as a single matrix and use the terms 'matrix' and 'image' interchangeably.

This section uses the following constants:

$\boldsymbol{X_i}$ refers to Input image, the image that is fed into the generator

$\boldsymbol{X_o}$ refers to the output image, the image that is returned by the generator

4.1.1. Motion Blur

Motion blur is a fairly common phenomenon in uncontrolled, outdoor environments and is typically a result of a camera being moved during image capture. However, it is also likely to occur in the presence of wind, where wind causes movement in the environment. This augmentation attempts to artificially recreate the effect of both naturally occurring wind and camera movement through the application of motion blur.

Unaugmented

Light Motion Blur

Property	Value
Direction	135
Size	5

Heavy Motion Blur

Property	Value
Direction	287
Size	25

Figure 4.2 Comparison of unaugmented input image (left) to output motion-blurred images (centre, right)

Table 4.1 Motion blur augmentation properties

Property	**Description**	**Range**
Direction	Angle of motion blur, measured in degrees from centre right of image	0 - 360
Size	Specifies the strength of the blur applied where strength is proportional to the value of *Size*. Value must be odd, even numbers are incremented by one.	1 - 25

The augmentation is achieved by creating a matrix $\boldsymbol{K}_{motionblur}$ of size *Size* which is rotated according to *Direction* and combined with the input image. This process is described by equations 4.1 - 4.5.

$$\boldsymbol{K}_{motionblur} = \begin{bmatrix} 1 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \iddots & \vdots \\ 0 & \cdots & 1 & \cdots & \vdots \\ \vdots & \iddots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 \end{bmatrix}$$

Equation 4.1 Creation of matrix $\boldsymbol{K}_{motionblur}$, a half-diagonal matrix of dimensions determined by Size

$\boldsymbol{K}_{motionblur}$ is dynamically created as a *Size*-by-*Size* half-diagonal matrix where the upper-left diagonal is populated with ones and the remainder of the main-diagonal with zeros.

$$\boldsymbol{R} = \begin{bmatrix} \alpha & \beta & (1-\alpha)\gamma - \beta\gamma \\ -\beta & \alpha & \beta\gamma + (1-\alpha)\gamma \end{bmatrix}$$

Equation 4.2 2D rotation transform matrix for warp affine transformation

A second matrix, $\boldsymbol{R}$, is created as shown in Equation 4.2 which is used to rotate the $\boldsymbol{K}_{motionblur}$ Rotation is achieved by performing a 'Warp affine' transformation. Warp affine is a transformational technique that can be used to distort a matrix whilst preserving the relational consistency between elements of the matrix. $\boldsymbol{R}$ is used to instruct the warp affine function perform a rotation, which is enabled by providing three variables: $\alpha = \cos(\textit{Direction})$, $\beta = \sin(\textit{Direction})$, and $\gamma = \frac{1}{2}\textit{Size}$.

$$\boldsymbol{K}_{motionblur}(x, y) = \boldsymbol{K}_{motionblur}(\boldsymbol{R}_{1,1}x + \boldsymbol{R}_{1,2}y + \boldsymbol{R}_{1,3},\ \boldsymbol{R}_{2,1}x + \boldsymbol{R}_{2,2}y + \boldsymbol{R}_{2,3})$$

Equation 4.3 Warp affine function, used to rotate a 2D matrix

Equation 4.3 uses the warp affine function to rotate $\boldsymbol{K}_{motionblur}$ by multiplying each value of $\boldsymbol{K}_{motionblur}$ by the sum of the first and second rows of $\boldsymbol{R}$.

$$\boldsymbol{K'}_{motionblur} = \frac{\boldsymbol{K}_{motionblur}}{\sum \boldsymbol{K}_{motionblur}}$$

Equation 4.4 Normalisation of matrix $\boldsymbol{K}_{motionblur}$

Once rotated, $\boldsymbol{K'}_{motionblur}$ is normalised as represented in Equation 4.4. Normalisation is important to ensure that when$\boldsymbol{K'}_{motionblur}$ is combined with the input image, the resultant pixel colours are within the same range and not either darker or brighter.

$$\boldsymbol{X}_o(x,y) = \sum_{i=-\gamma}^{\gamma} \sum_{j=-\gamma}^{\gamma} \boldsymbol{X_i}(x-i, y-j)\boldsymbol{K'}_{motionblur}(i,j)$$

Equation 4.5 Convolution function, for each element in $\boldsymbol{X}_i$, computes a new value using both $\boldsymbol{K'}_{motionblur}$ and surrounding elements.

In the final step, matrix $\boldsymbol{K'}_{motionblur}$ is convolved with the input image, represented as matrix $\boldsymbol{X}_i$. For each element, a calculation involving both the selected elements neighbouring elements and the elements of $\boldsymbol{K'}_{motionblur}$ is performed to determine the new value for the selected element in the output image, represented as matrix $\boldsymbol{X}_o$. This calculation is shown in Equation 4.5.

Figure 4.2 shows two different configurations of the properties listed in Table 4.2 which have been used to apply a light and heavy motion blur to the unaugmented image.

4.1.2. Brightness and Darkness

The brightness and darkness augmentations are generated using the same code within the generation program, however for simplicity, are collectively referred to as the 'brightness augmentation'. This augmentation is the simplest of all augmentations recreated by in this thesis requiring the least difficulty to implement, however is also arguably the most commonly encountered in uncontrolled outdoor environments as the time, weather and camera position play a large part in determining the lighting level of each image. This augmentation attempts to recreate the effect of changing environmental lighting conditions by simply adjusting the brightness of an image.

Unaugmented

Brighter Image

Property	Value
Brightness	64

Darker Image

Property	Value
Brightness	-64

Figure 4.3 Comparison of unaugmented input image (left) to output brightened image (centre) and output darkened image (right)

Table 4.2 Brightness augmentation properties

Property	**Description**	**Range**
Brightness	Relative brightness change based on 0, where 0 represents no adjustment from input image	-255 – +255

$$\boldsymbol{X}_o = \boldsymbol{X}_i + \alpha$$

Equation 4.6 Scalar addition, Scalar α is added to input matrix $\boldsymbol{X}_i$ to create output matrix $\boldsymbol{X}_o$

Using OpenCV, the brightness augmentation was achieved in a single line of code which adds a scalar colour byte value defined by *Brightness* (α) to the input image, naturally restricting the range of *Brightness* between -255 and 255. A brightened (centre) and darkened (right) version of the input image (left) are shown in Figure 4.3.

4.1.3. Sun Flare

The sun flare augmentation is a variation of the brightness augmentation, however rather than recreating environmental lighting, which is typically consistent over an image, this augmentation attempts to create smaller regions of high intensity brightness within the image

as typically occurs when a camera captures the sun. An immediate limitation of this augmentation is that no computation is done to attempt to find suitable or realistic placement locations within the input image. Instead, flares are generated randomly under the assumption that internal reflection within a camera lens can account for flares that do not appear to be directly caused by the sun.

Unaugmented

Light Sun Flare

Property	Value
Intensity	40
MaxPoints	50
MinPoints	10
Rounds	5
Size	101
Temperature	5000

Heavy Sun Flare

Property	Value
Intensity	80
MaxPoints	100
MinPoints	50
Rounds	20
Size	80
Temperature	7000

Figure 4.4 Comparison of unaugmented input image (left) to output sun flared images (centre, right)

Table 4.3 Sun flare augmentation properties

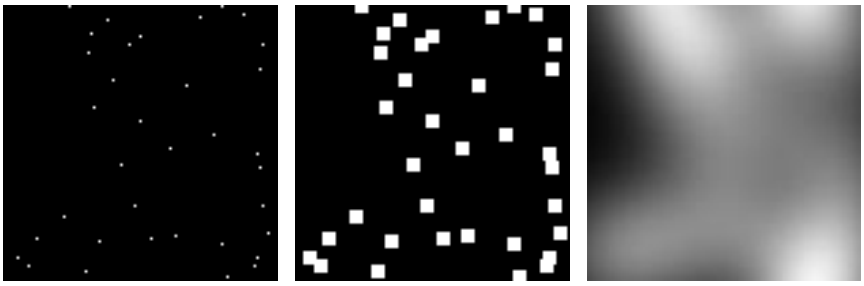
Property	**Description**	**Range**
Intensity	Alpha channel intensity, reducing this value makes the augmentation more translucent	0-100
MaxPoints	Maximum number of points that can participate in a single generation attempt, must be more than MinPoints	1-100
MinPoints	Minimum number of points that can participate in a generate attempt, must be less than MaxPoints	1-100
Rounds	Number of generation attempts. Each attempt is merged such that a higher number of attempts will create a stronger augmentation	1-50
Size	Size of matrix used to create augmentation	1-150
Temperature	Colour temperature in Kelvins	5000-7000

The final output of this augmentation can be seen in Figure 4.4 which shows both a light (centre) and heavy (right) sun flare augmentation created using two different configurations of the properties listed in Table 4.3 and applied to an unaugmented image (left).

$$\boldsymbol{M}_{sunflare} = \begin{bmatrix} 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix}$$

Equation 4.7 Creation of matrix $\boldsymbol{M}_{sunflare}$, a fully zeroed matrix of dimensions determined by Size

Generation of this augmentation begins by creating an empty matrix, $\boldsymbol{M}_{sunflare}$, as a *Size*-by-*Size* as shown in Equation 4.7. This matrix accumulates the output of each generation attempt (round) and is the matrix applied to the input image at the final step. The number of generation attempts is specified by the value of *Rounds*. The process undertaken by each generation attempt round can be split into three steps, as represented in Figure 4.5, these steps are explained below.



Step 1 Step 2 Step 3

Figure 4.5 Point transformation that occurs in each generation attempt within the sun flare augmentation

$$\boldsymbol{M}_{flare} = \begin{bmatrix} 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix}$$

Equation 4.8 Creation of matrix $\boldsymbol{M}_{flare}$, a fully zeroed matrix with the same dimensions as $\boldsymbol{M}_{sunflare}$

For each round, a new empty matrix, $\boldsymbol{M}_{flare}$, with the same dimensions as $\boldsymbol{M}_{sunflare}$ is created as shown in Equation 4.8.

Step 1 – Point randomisation. A random number of x, y coordinate points between *MaxPoints* and *MinPoints* are created with randomised x, y values chosen between 0 and *Size*. Each point generated represents an element in $\boldsymbol{M}_{flare}$ and is used to increase this value by *Intensity*. The result of this step is represented by the left-hand image within Figure 4.5 where a non-black, *Intensity*, coloured pixel indicates a randomly chosen coordinate point within $\boldsymbol{M}_{flare}$.

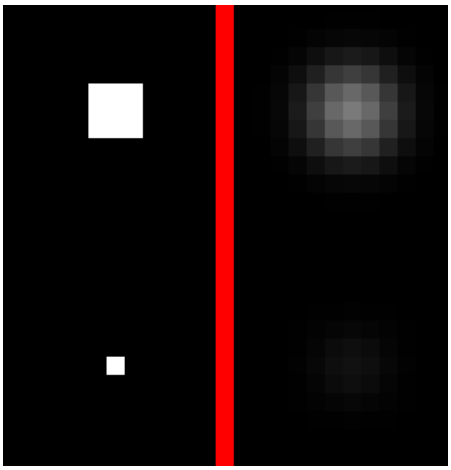


Figure 4.6 Effect of a 3x3 Gaussian blur on a 3x3 and 1x1 white pixel. Before blur (Left) and after blur (right)

Step 2 – Point enlargement. Following Step 1, each element of $\boldsymbol{M}_{flare}$ either has a value of zero, *Intensity*, or a multiple of *Intensity* in the event a point was chosen more than once. For the vast majority of non-zero elements, it can be assumed that its immediate neighbouring values were not randomly chosen and are of value zero. This presents a problem for step 3, which uses applies a Gaussian blur to $\boldsymbol{M}_{flare}$, as the overwhelming number of zero-valued elements effectively hides the non-zero value. To overcome this issue, each non-zero element is copied to its immediate neighbours using a 5x5 box filter which significantly reduces the number of zero-valued elements surrounding both the original non-zero, and newly created non-zero values. The effect of this step can be seen in Figure 4.6, which applies a 3x3 gaussian blur to the left-hand white pixels to create the right-hand blurred pixels. Post blur, both pixels are still present, however, the smaller white pixel has been reduced from an RGB value of 255

to 17 across each channel, making it effectively black and very difficult to see, the larger dot on the other hand while now a significantly darker colour, is still visible.

$$\boldsymbol{K}_{enlarge} = \frac{1}{25}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Equation 4.9 5x5 normalised box filter kernel, matrix $\boldsymbol{K}_{enlarge}$, used for enlarging non-zero values within $\boldsymbol{M}_{flare}$ [4]

The box filter operation is applied to $\boldsymbol{M}_{flare}$ by using the convolutional filter with a box filter kernel $\boldsymbol{K}_{enlarge}$. The box filter kernel defined in Equation 4.9 which creates a normalised, 5 by 5 matrix. Matrix $\boldsymbol{K}_{enlarge}$ is constant and not dependant on any of the augmentation's configurable properties. The convolution filter defined in Equation 4.5 can be reused to create a box filtered matrix, $\boldsymbol{M'}_{flare}$, with the following substitutions:

$\boldsymbol{X}_o$: $\boldsymbol{M'}_{flare}$, I: $\boldsymbol{X}_i$, and $\boldsymbol{K'}_{motionblur}$: $\boldsymbol{K}_{enlarge}$

Where the left-hand variable refers to Equation 4.5, and the right-hand variable a matrix from Section 4.1.3. The result of this step can be seen in the centre image of Figure 4.5.

$$\boldsymbol{K}_{Gaussian_{(x,y)}} = \frac{1}{2\pi\sigma^2} \times e^{\frac{(x-\frac{Size-1}{2})^2+(y-\frac{Size-1}{2})^2}{2\sigma^2}}$$

Equation 4.10 2D Gaussian blur kernel calculation [5]

$$\sigma = 0.3 \times \left(\frac{Size-1}{2} - 1\right) + 0.8$$

Equation 4.11 Gaussian blur standard deviation calculation

4 https://docs.opencv.org/3.4/d4/d86/group__imgproc__filter.html#gad533230ebf2d42509547d514f7d3fbc3, Accessed 25/04/2023

5 https://docs.opencv.org/3.4/d4/d86/group__imgproc__filter.html#gaabe8c836e97159a9193fb0b11ac52cf1

Step 3 – Smoothening. In Step 3, a Gaussian blur is applied to the matrix $\boldsymbol{M'}_{flare}$ created in Step 2 to create a Gaussian blurred matrix, $\boldsymbol{M''}_{flare}$. Equation 4.10 is used to calculate $\boldsymbol{K}_{Gaussian}$, the Gaussian blur kernel and depends on variables x, y, σ and *Size*. The variables x and y which are row and column coordinates starting at $x = 0, y = 0$ from the top left of $\boldsymbol{K}_{Gaussian}$ and have a maximum value of *Size*. The variable σ, which is the standard deviation, can be calculated using Equation 4.11, which also depends on *Size*.

As with the box filter, the Gaussian blur can be applied to matrix $\boldsymbol{M'}_{flare}$ to create the blurred matrix $\boldsymbol{M''}_{flare}$ by using the convolution filter defined in Equation 4.5 with the following substitutions:

$\boldsymbol{X}_o$: $\boldsymbol{M''}_{flare}$, $\boldsymbol{X}_i$: $\boldsymbol{M'}_{flare}$, and $\boldsymbol{K'}_{motionblur}$: $\boldsymbol{K}_{Gaussian}$

Where the left-hand variable refers to Equation 4.5, and the right-hand variable a matrix from Section 4.1.3. The result of this step can be seen in the left-most image of Figure 4.5.

$$\boldsymbol{M'}_{sunflare} = \frac{\boldsymbol{M}_{sunflare} + \boldsymbol{M''}_{flare}}{2}$$

Equation 4.12 Averaged matrix addition

After Step 3, the blurred matrix $\boldsymbol{M''}_{flare}$ is added to $\boldsymbol{M}_{sunflare}$ and divided by 2 as shown in Equation 4.12 to balance out the addition. For the next loop, $\boldsymbol{M}_{sunflare}$ becomes $\boldsymbol{M'}_{sunflare}$.

Once all rounds have been created and merged with $\boldsymbol{M}_{sunflare}$, another matrix, $\boldsymbol{M}_{suntemp}$ is created using the dimensions of the input image to represent a three-channel image where each element is set to the RGB representation of the value provided in *Temperature*. The code used to convert colour temperature into RGB can be found in Appendix F and was adapted from an online blog published by Neil Bartlett in 2015 [96]. $\boldsymbol{M}_{sunflare}$ is then resized and merged with $\boldsymbol{M}_{suntemp}$, adding an alpha channel and creating matrix $\boldsymbol{M'}_{suntemp}$.

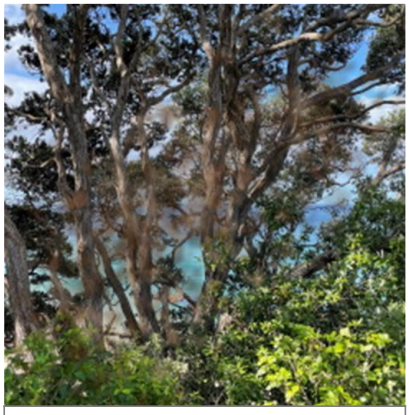
In the final step, Matrix $\boldsymbol{M'}_{suntemp}$ is overlaid onto the input image to create the output image.

4.1.4. Dirt

Dirt is a common issue within outdoor environments and can occur in a variety of forms. This augmentation looks to replicate two of these forms: Mud and dust. Mud is arguably the more anticipated form of dirt in outdoor environments and tends to stick on a camera and occlude large area of a captured image. Dust on the other hand typically collects on a camera overtime resulting in finer patches of occlusion.

Unaugmented

Light Dirt (Dust)



Property	Value
ClumpSize	40
ColourPalette	Palette2
DirtPatches	4
DoNotReusePoints	True
MaxDistance	400
MaxRounds	200
MinPoints	6
Opacity	128
Particles	2000
Scale	1.5
SmoothingAttempts	12
SmoothingIntensity	3

Heavy Dirt (Mud)

Property	Value
ClumpSize	40
ColourPalette	Palette1
DirtPatches	3
DoNotReusePoints	False
MaxDistance	400
MaxRounds	200
MinPoints	6
Opacity	64
Particles	1000
Scale	1.0
SmoothingAttempts	12
SmoothingIntensity	3

Figure 4.7 Comparison of unaugmented input image (left) to output dirtied images (centre, right)

Table 4.4 Dirt augmentation properties

Property	**Description**	**Range**
ClumpSize	Specifies the maximum number of particles that can be used to create a dirt clump polygon. Dirt clumps are combined to make dirt patches.	1-100
ColourPalette	Provides a collection of eight colours that are used to colour dirt clumps. See Appendix D to see palette colours and for more information on the ColourPalette type.	Palette1 / Palette2 / Palette3
DoNotReusePoints	If false, ensures that each particle is used in a maximum of one dirt clump polygon. If true, particles can be reused in multiple dirt clumps.	True / False
MaxDistance	Maximum distance to search for nearby particles when creating dirt clump polygons.	1-500
MaxRounds	Specifies the maximum number of times a particle can be "moved" when distributing the particles.	50-500
MinPoints	Specifies the minimum number of points a calculated dirt clump polygon must have to be considered a valid dirt clump. Invalid dirt clumps are not used for dirt patch creation.	3-100
Opacity	Specifies the opacity of each dirt clump polygon when rendered	0-255
Particles	Specifies the number of random points (particles) that are used each round to creation the individual dirt clumps in each dirt patch.	100-10000
DirtPatches	Specifies the number of dirt patches to generate on the image	1-5
Scale	Specifies the floating point multiplier used to resize the final augmentation	0.25-2.0
SmoothingAttempts	Number of smoothing operations to apply. A higher number of blurs should smoothen the edges of dirt patches. Works in conjunction with *SmoothingIntensity*.	0-50
SmoothingIntensity	Strength of blur used to smooth dirt patches on each smoothing attempt. Works in conjunction with *SmoothingAttempts*.	1-10

The dirt augmentation is comprised of *DirtPatches* number of dirt patches which are each created using the five steps described below. Each dirt patch is generated in on its own matrix, $\boldsymbol{M}_{dirtpatch}$, which has a width and height of $244 \times Scale$. Creating the matrix in this manner enables the dirt augmentation to be applied to any image, regardless of size, and has the advantage that there is less dependence on the image size when determining which configurable property values to use. Figure 4.8 shows the intermediate output of each step.

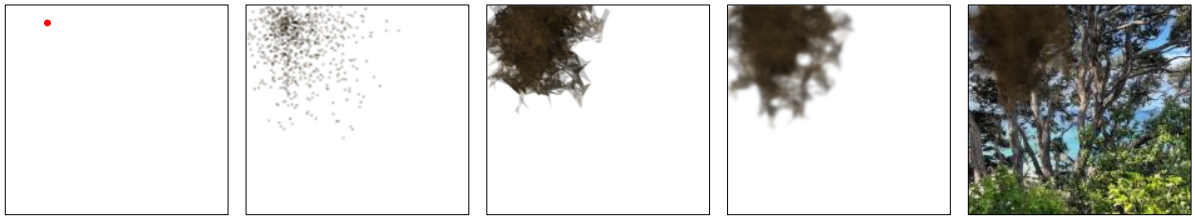


Figure 4.8 Dirt clump creation steps

Step 1. An origin point is chosen somewhere inside $\boldsymbol{M}_{dirtpatch}$, in Figure 4.8 the origin point is represented by a red dot. This origin point becomes the centre of the dirt patch being generated.

Step 2. A number of points, as specified by *Particles*, are generated around the origin in such a manner that the point density is higher closer to the origin. The points are treated as particles in order to reproduce dirt being 'thrown' onto the camera. This method helps to achieve a more consistent natural distribution of points in comparison to generating point clusters around an origin. To achieve this, a particle has both a random weight and velocity which, over a period of *MaxRounds*, can be used to simulate particle movement and distribute points over the image. In early attempts, particle collision was modelled to allow for more natural interaction between particles, however experiments showed that a natural distribution could be achieved without particle collision or realistic velocity calculations. To achieve the distribution shown in step 2 of Figure 4.8, a virtual 3D Box with a top-down width and height matching $\boldsymbol{M}_{dirtpatch}$ and a depth of 100 is created to simulate particle movement. A fixed number of *Particles* particles are created as a randomised three-dimensional point with two additional values for speed and weight. For each particle, the x, y coordinates represent a randomised location within the bounds of $\boldsymbol{M}_{dirtpatch}$, the z value represents the depth of the world and is set to 100, the speed

is randomised between 10 and 40 and influences the rate of movement in all directions, and the weight is assigned a random number between 1 and 5 and influences the particles stopping power in all directions. When placed in the world, each particle is given the coordinate of the origin point which is combined with the speed and weight values to simulate velocity. Particles are moved once per round for a maximum of *MaxRounds*. In each round, new directions for each of the x, y, z coordinates are calculated depending on whether or not they have hit the origin. If the particle is yet to reach the origin, the new direction is calculated to move the particle towards the origin. Otherwise, the particles direction does not change until it hits the worlds bounds where its speed is divided by three times its weight and reflected. The computed direction for each particle is then multiplied by its speed and weight to create a velocity.

After all rounds are completed, the x, y values of each particle are collected as points and are ready for the next step. The pseudo code for this step is provided in Appendix E.

Step 3. The generator attempts to create 'dirt clumps', which are effectively polygons, using the points generated in Step 2. The properties *ClumpSize*, *DoNotReusePoints*, and *MaxDistance* are used to attempt creation of *Particles* dirt clumps, ensuring that each point has the opportunity to become a candidate for a dirt clump. Dirt clumps are created by choosing the nearest *ClumpSize* points that are within *MaxDistance* and if *DoNotReusePoints*, unused. This process results in up to *Particles* number of clumps which are never re-ordered to prevent self-intersection. Once all clumps have been created, a check is performed removing all dirt clumps that contain fewer than *MinPoints* points.

The remaining dirt clumps are rendered using a colour created by combining *Opacity* and a random colour from *ColourPalette*. Three dirt colour palettes were created for this project, all a collection of eight colours that have been selected from images containing natural dirt. The colours in each palette are provided in Appendix D.

Step 4. A Gaussian blur of *SmoothingIntensity* is applied *SmoothingAttempts* times. This is achieved by using Equations 4.10 and 4.11 where *Size* = *SmoothingIntensity* on $\boldsymbol{M}_{dirtpatch}$. This process helps to smooth the edges of the dirt patches in addition to mixing the layered colours in the denser areas. It was found that applying many small blurs created a much smoother transition between the layered dirt clumps and the background, hence it is recommended that a high number of low-intensity blurs are favoured over other combinations.

Step 5. Finally, $\boldsymbol{M}_{dirtpatch}$ is resized to match the size of the input image $\boldsymbol{X}_i$ and rendered onto the input image. At this stage, the contents of $\boldsymbol{M}_{dirtpatch}$ are considered to be a single dirt patch.

4.1.5. Water-based Augmentations

This section focuses on the final four augmentations which are all water-based. Each of these four augmentations are implemented using a common codebase before being 'specialised' to achieve their desired augmentation. These augmentations ultimately perform the same initial steps and are therefore described together in this section.

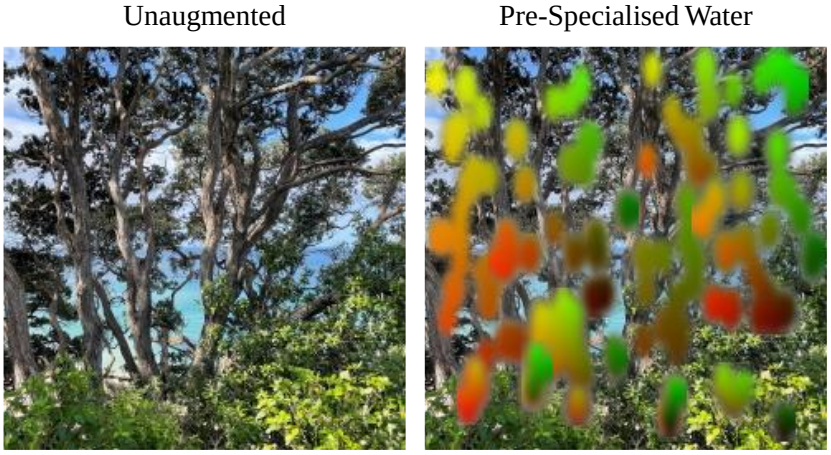


Figure 4.9 Comparison of unaugmented input image (left) to pre-specialised water-based augmentation (right)

Table 4.5 pre-specialised water-based augmentation properties

Property	**Description**
Drops	Number of waterdrops to use to generate the augmentation (Note, water-drops are grouped to create raindrops, so this property does not represent the final number of raindrops present)
Refraction	Specifies the padded region behind each raindrop which is used to create the refraction effect
Tolerance	Maximum distance between waterdrops allowed when merging into raindrops

Figure 4.9 shows the output of the initial shared steps for the final augmentations, the output image in this case is a collection of multi coloured raindrops which, on close inspection, are grouped with some of their neighbouring raindrops. The aim of this initial shared process is to replicate naturally occurring raindrops as they would appear on a transparent protection window in front of the camera lens, with a specific focus on the grouping merging of close raindrops. Table 4.5 lists the common properties required to implement the shared code.

Step 1. *Drops* number of rectangles, representing waterdrops, are created at a random location within the bounds of the image's size. These waterdrops are created on a matrix $\boldsymbol{M}_{drops}$ which is of the same dimensions as the input image $\boldsymbol{X}_i$. Each rectangle has a randomised size between a configurable lower and upper bound which is determined individually by each augmentation for scaling flexibility and will be discussed later in this section.

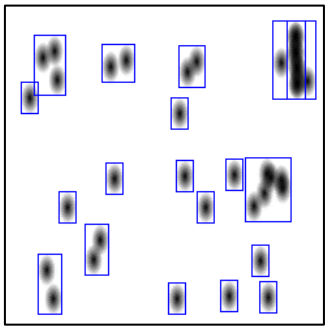


Figure 4.10 A collection of black waterdrops with blue rectangles indicating which waterdrops are considered neighbours

Step 2. Once each rectangle is created, the rectangles are merged. Rectangle collision is calculated by looping over all rectangles sequentially in a 2-level nested loop, inflating both targets by *Tolerance* and checking for intersection. If the rectangles intersect, the rectangle belonging to the inner nest is made a child of the rectangle in the outer nest. Any rectangle in that becomes a child of another rectangle is ignored for the remainder of the merging process.

Once merging is complete, the complete bounds for each merged rectangle are computed and each rectangle is rendered using a stretched 'alpha' drop, which is an image with an opaque black pixel in the centre that radially loses opacity towards the edges of the image. Figure 4.10 shows the computed bounds in blue of each drop considered to be neighbours.

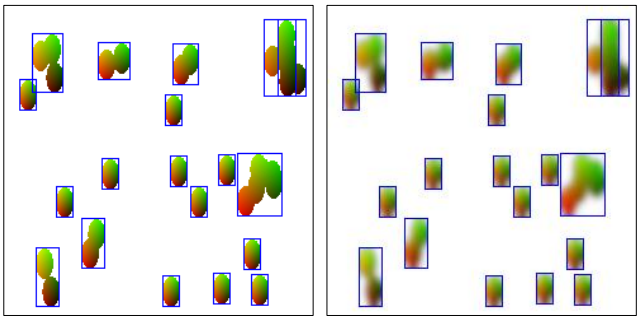


Figure 4.11 Result of normalisation and 'blobification' of raindrops

Step 3. For simplicity, each collection of ellipses bounded by a blue rectangle is considered a single raindrop going forward. Each of these raindrops are moved to their own matrix, generically referred to as $\boldsymbol{M}_{raindrop}$, which sized to match the dimensions of its respective raindrops complete bounds as indicated by the blue rectangles.

In this step, two actions are performed for every raindrop. The first action is normalisation which re-colours the black ellipses within each $\boldsymbol{M}_{raindrop}$ by overlaying it with the normal map presented in Figure 4.12. The re-colouring process preserves the alpha channel such that the resulting raindrops now render as shown in the left-hand side image in Figure 4.11. This action is important for attempting refraction, which is discussed in a later step.

Figure 4.12 Raindrop normal map

The second action is to ‘blobify’ the waterdrops that makeup each raindrop by merging nearby waterdrops such that they form a more natural state. To achieve this, an attempt was made to recreate the “The Gooey Effect” [97] using OpenCV.

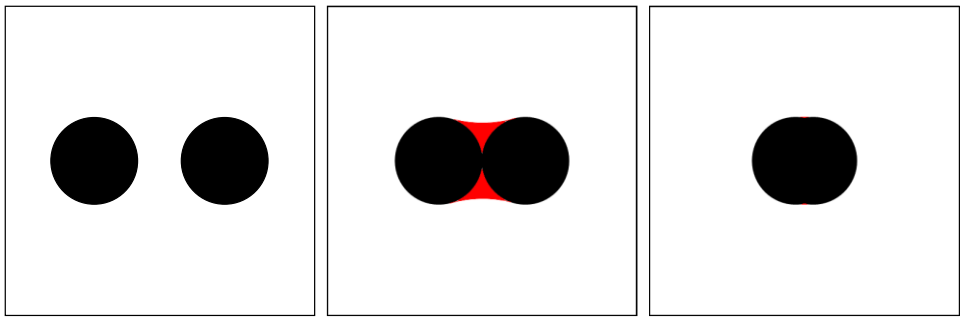


Figure 4.13 Example of how the gooey effect tries to merge with nearby objects based on proximity

The Gooey Effect is a shape blobbing technique designed for website CSS designs and implemented using a combination of SVG filters. This technique is designed to be used in CSS animations when merging non-rectangular objects to create a more seamless transition. Figure 4.13 shows three panels each with two objects with different gap sizes between them. In the figure, the red areas represent the extra pixels that are added through the technique to make each objects appear to be integrating with each other if they are in close enough proximity. This technique is achieved by applying both a Gaussian blur and a special convolution operation that emulates an SVG colour matrix to $\boldsymbol{M}_{raindrops}$ to create $\boldsymbol{M'}_{raindrops}$.

$$\boldsymbol{M}_{colourmatrix} = \frac{1}{13}\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 19 & -9 \end{bmatrix}$$

Equation 4.13 Creation of $\boldsymbol{K}_{\boldsymbol{colourmatrix}}$. A Normalised 5 by 4 colour matrix

$$C' = \begin{pmatrix} C'_r \\ C'_g \\ C'_b \\ C'_a \end{pmatrix} = \begin{pmatrix} C_r Y_{1,1} + C_g Y_{1,2} + C_b Y_{1,3} + C_a Y_{1,4} + Y_{1,5} \\ C_r Y_{2,1} + C_g Y_{2,2} + C_b Y_{2,3} + C_a Y_{2,4} + Y_{2,5} \\ C_r Y_{3,1} + C_g Y_{3,2} + C_b Y_{3,3} + C_a Y_{3,4} + Y_{3,5} \\ C_r Y_{4,1} + C_g Y_{4,2} + C_b Y_{4,3} + C_a Y_{4,4} + Y_{4,5} \end{pmatrix}$$

Equation 4.14 Per-pixel RGBA colour matrix transformation

Equation 4.13 provides the normalised colour matrix, $\boldsymbol{K_{colourmatrix}}$ used in this augmentation. Adapted from [97] and is designed such that each row provides a set of four channel multipliers and a constant that is used by SVG's colour filter to create a new colour for each existing image pixel as shown in Equation 4.14 where $\boldsymbol{C}$ is the RGBA input colour, $\boldsymbol{C'}$ is the output colour and $\boldsymbol{Y = M}_{colourmatrix}$.

The Gooey Effect uses this transformation to reduce the transparency of 'blurred edges' added by the preceding Gaussian blur without modifying the RGB channels, effectively adding the red areas as shown in Figure 4.13.

After completion of step 3, each instance of $\boldsymbol{M'}_{raindrop}$ can be overlaid over the input image to achieve the augmentation as shown in Figure 4.9. From this point on, all four augmentations take different approaches in generating their respective augmentations making the collection of $\boldsymbol{M'}_{raindrop}$ the texture-normalised matrices that are passed to each of the water-based augmentations for specialisation, however a common theme they all share is usage of refraction.

Refraction is responsible for the distortion effect experienced when looking through raindrops. In this project, refraction is recreated by attempting to re-colour each raindrop with an inverted and rescaled duplicate of the pixels behind each raindrop ($\boldsymbol{M}_{refractdrop}$).

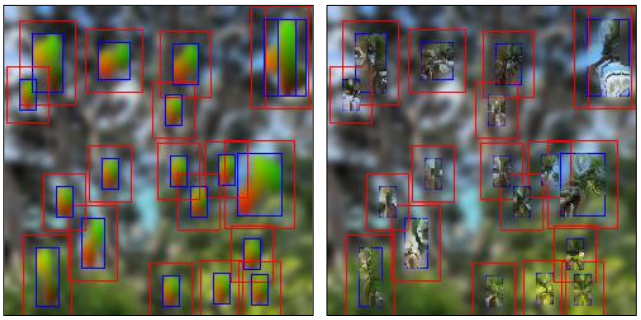


Figure 4.14 Usage of normal mapping to translate refracted area behind raindrops onto each raindrop

The normal map shown in Figure 4.12 is a 256 x 256 pixel image specially designed such that each sequential column and row reduces their red and green channels respectively by one. Taking advantage of every single pixel in this image having a unique colour, this technique is commonly used to encode x and y coordinates. In Figure 4.14, the *Refraction* property is used to create a red bounding rectangle around each raindrop. The area defined by each red rectangle represents the collection of refraction pixels that will be downscaled to 256 by 256 pixels and inverted along both x and y coordinates. Using the red and green values applied to the raindrops, the raindrops can be re-coloured using the pixel values of $\boldsymbol{M}_{refractdrop}$.

Raindrop scaling relative to the size of the image presents an obstacle for the generation of water-based augmentations as unlike the non-water-based augmentations discussed so far, the water-based augmentations can easily look out of place when incorrectly scaled. Scaling can be achieved in two ways, the first is to provide configurable properties that can be adjusted based on the size of a provided image, and the second is to attempt to automatically adjust raindrop size proportionally with the size of the image. In this thesis, the latter approach was implemented by using the following two equations shown in Equation 4.15 and Equation 4.16 where w_s, h_s, w_o, h_o are scaled width, scaled height, original width and original height respectively.

Equation 4.15 Width scaling equation

$$w_s = a * e^{bw_o}$$

Equation 4.16 Height scaling equation

$$h_s = c * e^{dh_o}$$

Each water-based augmentation uses different constants a, b, c, and d (See Appendix G) which were determined by finding the optimum raindrop sizes at three different resolutions and finding a mathematical relationship between them. This method is better than requiring manual configuration of optimum raindrop size of datasets that do not use a consistent image size.

4.1.5.1. Focused Raindrops

The focused raindrop augmentation attempts to replicate raindrops as they would appear on a camera's protection window with the raindrops in focus.

Unaugmented

Light Focused Raindrops

Property	Value
Drops	100
Refraction	50
Tolerance	10
BackgroundBlur	5
ForegroundBlur	1

Heavy Focused Raindrops

Property	Value
Drops	500
Refraction	100
Tolerance	5
BackgroundBlur	10
ForegroundBlur	1

Figure 4.15 Comparison of unaugmented input image (left) to output images (centre, right) with in-focus raindrops applied

Table 4.6 Focused raindrops augmentation properties

Property	**Description**	**Range**
Drops	Number of waterdrops to use to generate the augmentation (Note, water-drops are grouped to create raindrops, so this property does not represent the final number of raindrops present)	0-1000
Refraction	Specifies the padded region behind each raindrop which is used to create the refraction effect	0-100
Tolerance	Maximum distance between waterdrops allowed when merging into raindrops	0-10
BackgroundBlur	Intensity of defocusing blur applied to background	1-25
ForegroundBlur	Intensity of blur applied to raindrop refraction	1-25

Table 4.6 lists the properties available for configuration of this augmentation alongside the values used to achieve the augmentation shown in Figure 4.15. This augmentation was created using the process described in Section 4.1.5, however, it also blurs applies a Gaussian blur to the input image of an intensity specified by *BackgroundBlur*. Background blurring is performed under the assumption that if the camera is focused on the foreground raindrops, the background will be out of focus. Both foreground and background blurring is achieved using Equation 4.5 to apply a custom Gaussian kernel ($\boldsymbol{K'}_{Gaussian}$) created using Equations 4.10, and 4.11 where either *Size* = *BackgroundBlur* or *Size* = *ForegroundBlur*.

4.1.5.2. Unfocused Raindrops

The unfocused raindrop augmentation attempts to replicate raindrops as they would appear on a camera's protection window with the raindrops out of focus.

Unaugmented

Light Unfocused Raindrops

Property	Value
Drops	25
Refraction	0
Tolerance	10
ForegroundBlur	1
Intensity	32

Heavy Unfocused Raindrops

Property	Value
Drops	150
Refraction	20
Tolerance	10
ForegroundBlur	1
Intensity	64

Figure 4.16 Comparison of unaugmented input image (left) to output images (centre, right) with unfocused raindrops applied

Table 4.7 Unfocused raindrops augmentation properties

Property	**Description**	**Range**
Drops	Number of waterdrops to use to generate the augmentation (Note, water-drops are grouped to create raindrops, so this property does not represent the final number of raindrops present)	25-1000
Refraction	Specifies the padded region behind each raindrop which is used to create the refraction effect	0-20
Tolerance	Maximum distance between waterdrops allowed when merging into raindrops	0-10
ForegroundBlur	Intensity of blur applied to raindrop refraction	0-10
Intensity	Intensity of blur applied to foreground (Applied after all raindrops are rendered)	32-64

The unfocused raindrops augmentation was created using the same process as described in Section 4.1.5 and is a small adjustment to the focused raindrops augmentation. The difference in this augmentation is that rather than blurring the background to create the illusion of the foreground being in focus, the foreground is blurred creating the illusion of the background being in focus. The intensity of the applied foreground blur is determined by *Intensity*. As with the focused raindrops, both foreground blurs are achieved using Equation 4.5 to apply a custom Gaussian kernel ($\boldsymbol{K'}_{Gaussian}$) created using Equations 4.10, and 4.11 where either *Size = ForegroundBlur* or *Size = Intensity*.

4.1.5.3. Frost

The frost augmentation is an intensified version of the focused raindrops augmentation which renders large overlapping drops which create a glass like frost effect. In many climates, frost is a common occurrence in winter resulting from the cooling of atmospheric moisture. When frost forms on a camera's protection window, the thin layer of ice can distort captured images.

Unaugmented

Light Frost

Property	Value
Drops	700
Refraction	1
Tolerance	2

Heavy Frost

Property	Value
Drops	1000
Refraction	5
Tolerance	5

Figure 4.17 Comparison of unaugmented input image (left) to output images (centre, right) with unfocused raindrops applied

Table 4.8 Frost augmentation properties

Property	Description	Range
Drops	Number of waterdrops to use to generate the augmentation (Note, water-drops are grouped to create raindrops, so this property does not represent the final number of raindrops present)	200-1000
Refraction	Specifies the padded region behind each raindrop which is used to create the refraction effect	1-5
Tolerance	Maximum distance between waterdrops allowed when merging into raindrops	0-10

This augmentation relies heavily on the waterdrop merging performed in the common code to generate large raindrops. This can be better seen in left-hand image of Figure 4.18 which shows the merged raindrops before performing refraction. Notably, this augmentation does not require any additional parameters beyond what is needed to perform the common steps.

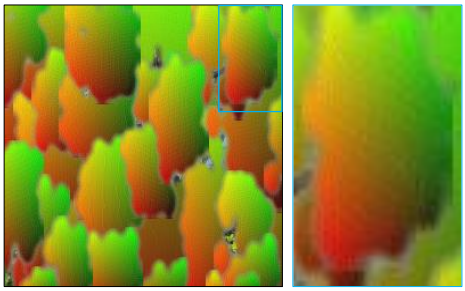


Figure 4.18 Frost augmentation with refraction re-colouring disabled showing the shapes of generated raindrops (left) and the resultant normalised texture pattern (right)

Each raindrop is created by a large number of smaller merged waterdrops. To ensure waterdrops are generated close enough to create large raindrops, this augmentation requires usage of a minimum of 700 for *Drops*. To balance the recommended increase in *Drops*, smaller sized waterdrops are used by using a smaller set of scaling constants for step 1 of Section 4.1.5. These constants can be found in Appendix G.

$$\boldsymbol{K}_{sharpenfrost} = \begin{bmatrix} -2 & -1 & -2 \\ -1 & 13 & -1 \\ -2 & -1 & -1 \end{bmatrix}$$

Equation 4.17 Creation of sharpening kernel for frost augmentation, used to create frost patterns

There is a notable difference between the texture of the normalised raindrops shown in right-hand, zoomed in image in Figure 4.18 and the texture in other normalised figures where the raindrops in Figure 4.18 contain a fingerprint-like pattern. This difference occurs as a result of passing each raindrop through an additional convolution operating using the sharpening kernel in Equation 4.17 directly after 'blobifying' the raindrops which creates the illusion of ice edges within each raindrop. Sharpening is typically used to increase the contrast between two neighbouring colours by using a normalised kernel to amplify the differences between nearby colours. In theory, at standard resolution of 255x255 pixels, the normal texture has a consistent difference of 1 between its cardinal neighbours meaning that sharpening should have no noticeable effect. However, upon closer inspection of the normal texture used for this project, a number of minute anomalies can be found where the difference between a pixel's cardinal neighbour is not 1. These anomalies do not interfere with normalisation as during creation of the raindrops, the normal texture is both stretched and blurred introducing further anomalies that would be amplified by the sharpening matrix. This pattern is created through the combination of existing anomalies in the normal texture, per-raindrop stretching, and a strong sharpening kernel. The effect of using a non-consistent normal texture can be seen in Figure 4.19. Note, the pattern present in the consistent normal texture is likely as a result of the normal texture being stretch to match the dimension of each raindrop, introducing inconsistencies that the sharpening operation will amplify.

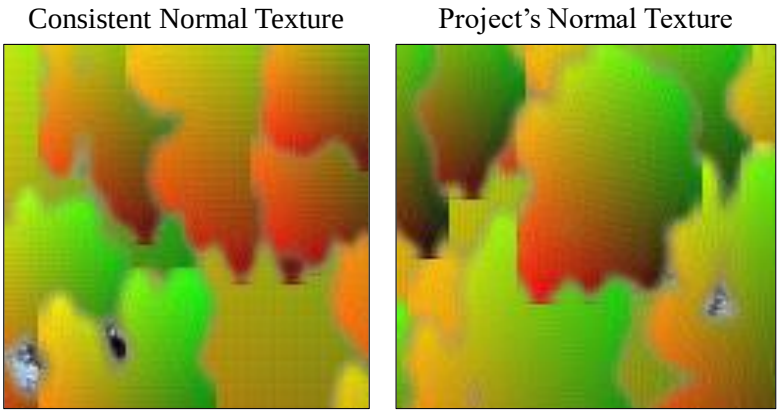


Figure 4.19 Comparison of the fingerprint like pattern between a consistent normal texture and the normal texture used by this project

$\boldsymbol{K}_{sharpenfrost}$ is a particularly strong kernel which is evident from the numerical difference between the centre value, 13, which represents the pixel being manipulated by the kernel during convolution, and its' neighbouring values. However, with the exception of using a using a sharpening matrix that ignores diagonal neighbours which resulted in a weaker sharpening effect, no notable difference was identified as a result of using different values for $\boldsymbol{K}_{sharpenfrost}$. $\boldsymbol{K}_{sharpenfrost}$ was applied as a convolutional filter to each raindrop ($\boldsymbol{M'}_{raindrop}$) using Equation 4.5 where $\boldsymbol{X}_i = \boldsymbol{M'}_{raindrop}$, $\boldsymbol{X}_o = \boldsymbol{M}_{frostdrop}$, and $\boldsymbol{K'}_{motionblur} = \boldsymbol{K}_{sharpenfrost}$ creating an instance of $\boldsymbol{M}_{frostdrop}$ for each instance of $\boldsymbol{M'}_{raindrop}$.

Next each $\boldsymbol{M}_{frostdrop}$ is refracted as described in Section 4.1.5 and layered underneath a cloned copy of itself which has had all pixels with an alpha channel value greater than 60 replaced with a translucent white colour, creating $\boldsymbol{M'}_{frostdrop}$. This step adds the slight hazy white tint commonly associated with frost.

The centre and right-hand images of Figure 4.17 show the resultant augmentation when all instances of $\boldsymbol{M'}_{frostdrop}$ are merged with the input image, $\boldsymbol{X}_i$.

4.1.5.4. Water Residue

The final augmentation created was the water residue augmentation. This augmentation was intended to recreate the watermark residue that often remains after raindrops have evaporated off glass.

Unaugmented

Light Water Residue

Property	Value
Drops	100
Refraction	0
Tolerance	50

Heavy Water Residue

Property	Value
Drops	300
Refraction	40
Tolerance	90

Figure 4.20 Comparison of unaugmented input image (left) to output images (centre, right) with water residue applied

Table 4.9 Water residue augmentation properties

Property	**Description**	**Range**
Drops	Number of waterdrops to use to generate the augmentation (Note, water-drops are grouped to create raindrops, so this property does not represent the final number of raindrops present)	50-300
Refraction	Specifies the padded region behind each raindrop which is used to create the refraction effect	0-40
Tolerance	Maximum distance between waterdrops allowed when merging into raindrops	50-90

The augmentation shown in Figure 4.20 appears visually similar to the unfocused rain augmentation shown in Figure 4.16. However, the image appears whiter and lacks adjustable foreground blurring. The resultant augmentation shown in Figure 4.20 was created using the

property values as specified in Table 4.9. Notably, this augmentation does not provide any additional configurable properties beyond what is required for implementation of the common steps. This augmentation uses a combination of small raindrops, light blurring, and high transparency to create a smudge like effect.

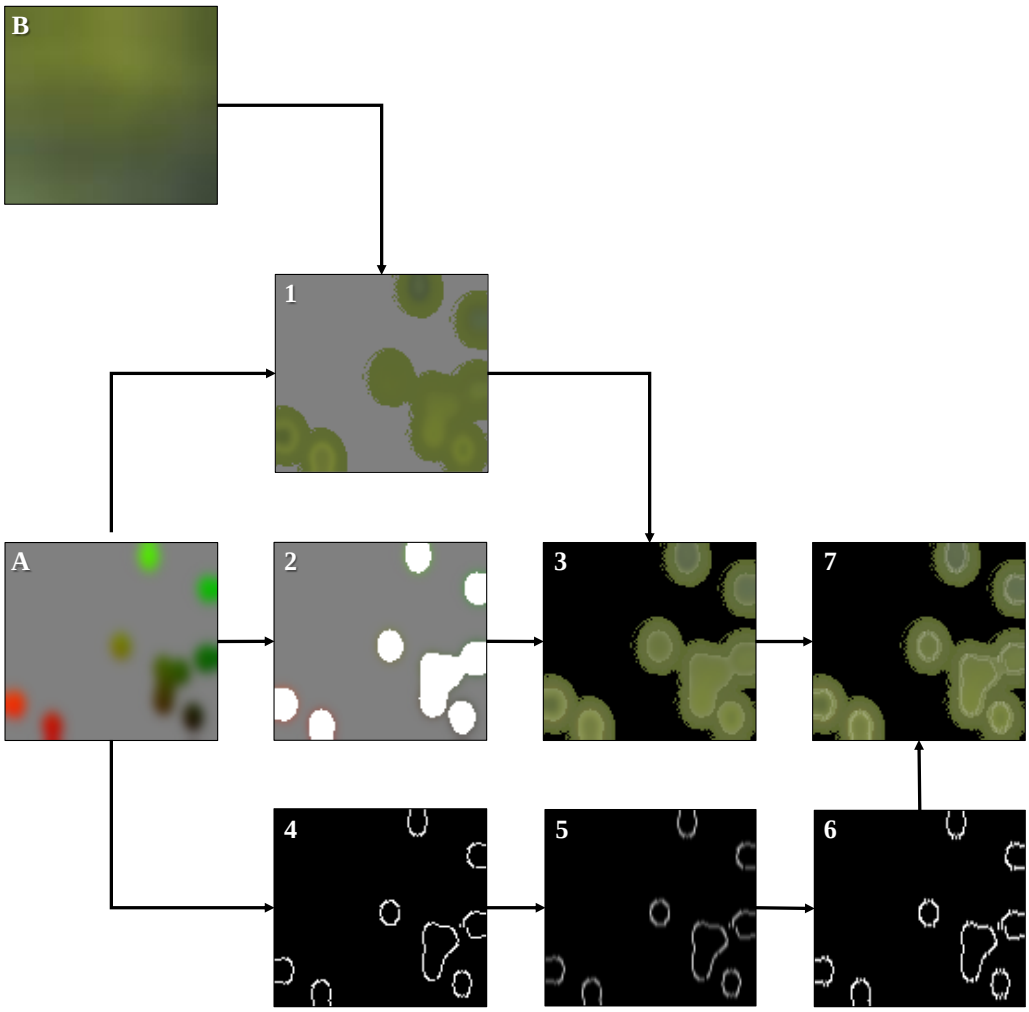


Figure 4.21 Water residue creation process steps

Figure 4.21 contains seven steps and two inputs representing the process chain that is applied to each instance of $\boldsymbol{M'}_{raindrop}$ and used to create the water residue augmentation. In this figure, Image A shows $\boldsymbol{M'}_{raindrop}$, the normal-textured result of a single raindrop created in Section 4.1.5, and Image B, the refracted input image $\boldsymbol{M}_{refractdrop}$ of each raindrop which is generated as per Section 4.1.5. The arrows in this figure are used to show whether an image has been cloned, transformed, or merged. Following these arrows, there are two distinct operations that have been performed on Image A which are later merged in Step 5.

Step 1. A duplicate of Image A is merged with Image B by replacing colours in Image A with its respective pixel in Image B where the colour in Image A has an alpha value greater than 0.

Step 2. A mask is created of Image A that replaces pixels with an alpha channel greater than 60 with a constant translucent white colour with a transparency of 80.

Step 3. The mask created in Step 2 layered on top of the output of Step 1 creating $\boldsymbol{M'}_{refractdrop}$. This step uses the mask to create the effect of dried-up water.

Step 4. The border of each waterdrop inside Image A is found using Canny edge detection to create $\boldsymbol{M}_{canny}$. Canny edge detection is used to detect areas of rapid intensity changes through a multistep process and generate an output result of black and white pixels where white indicates detected edges. Full details of canny edge detection are given in [98, 99] as further discussion of this approach is outside the scope of this thesis.

Step 5. $\boldsymbol{M}_{canny}$ is blurred using the convolution function in Equation 4.5 with a randomised Gaussian kernel created by Equations 4.10 and 4.11 where *Size* is a random number between 1 and 25. This step is performed in preparation for Step 6 and creates $\boldsymbol{M'}_{canny}$. The use of a randomised Gaussian blur varies the thickness between other instances of $\boldsymbol{M'}_{raindrop}$ after Step 6 such that after each instance of $\boldsymbol{M'}_{raindrop}$ has been passed through steps 1 – 7, the final augmentation will have water residue with different thicknesses.

Step 6. The output of Step 5, $\boldsymbol{M'}_{canny}$, is passed through the convolution function again where $\boldsymbol{K'}_{motionblur} = \boldsymbol{M'}_{canny}$, $\boldsymbol{X_i} = \boldsymbol{M'}_{canny}$, and $\boldsymbol{X_o} = \boldsymbol{M''}_{canny}$. This step strengthens the edges weakened as a result of blurring whilst also making them slightly thicker and more prominent than before blurring.

Step 7. Finally, $\boldsymbol{M''}_{canny}$ and $\boldsymbol{M'}_{refractdrop}$ are merged creating $\boldsymbol{M'}_{residuedrop}$, the water residue effect. The thickened edges created in steps 4-6 help the existing edges in $\boldsymbol{M'}_{refractdrop}$ standout whilst also making them appear more jagged and crusty.

The centre and right-hand images of Figure 4.20 show the resultant augmentation when all instances of $\boldsymbol{M'}_{residuedrop}$ are merged with the input image, $\boldsymbol{X}_i$.

4.2. Modified Datasets

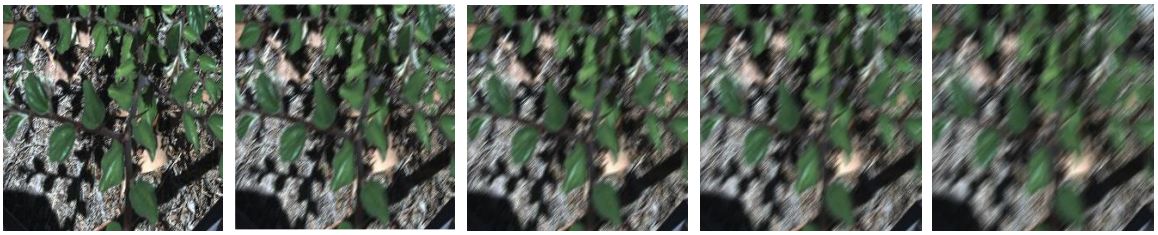
This section discusses the application of each of the nine augmentations to each of the datasets. For each augmentation, five pre-sets from lowest intensity (1) to highest intensity (5) were systematically created to provide an increasing level of intensity to cover the range of possible intensities that could be expected to be encountered in an uncontrolled outdoor environment. Taking advantage of the bulk pre-sets described in Section 4.1, intensities could be created to allow for small a degree of randomisation, ensuring each augmentation was unique without generating augmentations of either lower or higher intensities. As most augmentations are capable of scaling their output based on the size of the input image, the majority of the augmented datasets discussed in this chapter could be generated using the same pre-set configurations. The exceptions to this were the focused and unfocused rain augmentations. As a result of the type of effect, image sizes, and perceived distances between camera and object, these two augmentations required slightly different property configurations for each dataset.

4.2.1. Pre-sets

Figure 4.22 Unaugmented member image of the DeepWeeds dataset provided as a unaugmented comparison for augmented figures in this chapter. Adapted from [100] (Apache 2.0 License)

Figure 4.22 contains the unaugmented DeepWeeds dataset image referenced throughout this section.

Motion blur. The Motion blur augmentation has only two configurable properties, *Size* and *Direction* which are defined in Table 4.1. The *Direction* property is configured such that a random direction is picked for each image and is not altered across the intensities. The *Size* property however increases over sequential intensities such that all intensities have a range of 4 and that the upper bound of each intensity becomes the lower bound of the next intensity. The property range used for this pre-set are provided in Table 4.10, and were configured to produce the five intensities as shown in Figure 4.9



Intensity 1 Intensity 2 Intensity 3 Intensity 4 Intensity 5

Figure 4.23 Sample of the 5 Intensities of Motion Blur Pre-set applied to Figure 4.22

Table 4.10 Motion Blur Pre-Set values as configured for intensities 1-5

INTENSITY		**1**		**2**		**3**		**4**		**5**	
PROP	**TYPE**	**min**	**max**	**min**	**max**	**min**	**max**	**min**	**max**	**min**	**max**
Direction	**I**	0	360	0	360	0	360	0	360	0	360
Size	**I**	0	4	4	8	8	12	12	16	16	24

Although the *Size* property is listed as being configurable up to 99, a range of 0 – 24 was deemed to be a more realistic representation of what would be expected to occur naturally within an uncontrolled, outdoor environment.

Brightness and Darkness. Both augmentations have been merged into the same pre-set configuration as a result of sharing both the same properties with the only difference being the use of negative values for dark images and positive for bright. Naturally, for bright images, the *Brightness* property is restricted to 0 – 255 and for dark images -255 – 0. Each intensity has been assigned a range of 25 such that all five intensities offer brightness adjustments from 0 – 128 or -128 – 0 for bright and dark augmentation respectively. The property ranges used for

each intensity of this pre-set are provided in Table 4.11 and Table 4.12, and were used to produce the intensity samples as shown in Figure 4.24.

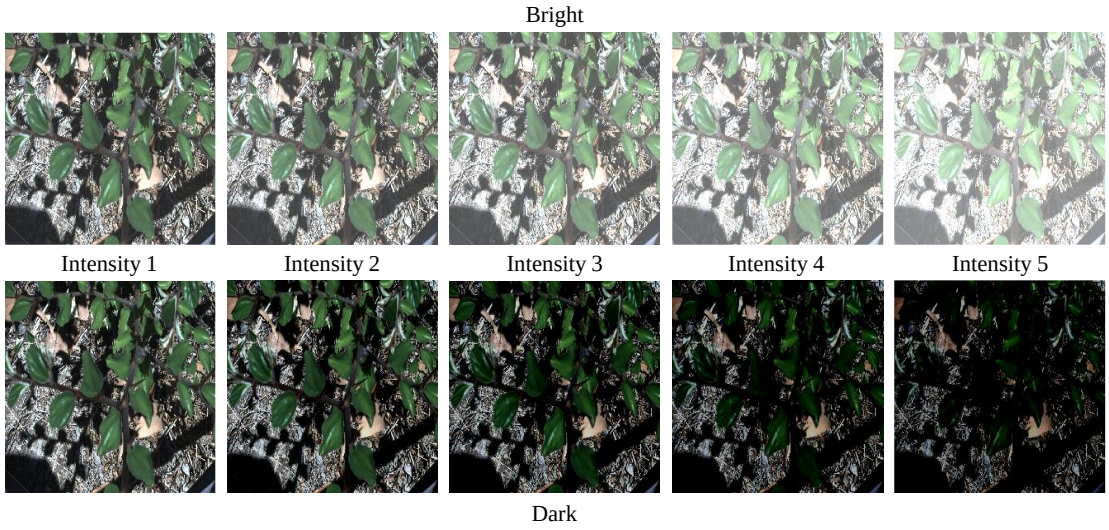


Figure 4.24 Sample of the 5 Intensities created for both bright and dark augmentations applied to Figure 4.22

Table 4.11 Darkness pre-set values as configured for intensities 1-5

INTENSITY		**1**		**2**		**3**		**4**		**5**	
PROP	**TYPE**	**min**	**max**	**min**	**max**	**min**	**max**	**min**	**max**	**min**	**max**
Brightness	**I**	-24	0	-49	-25	-74	-50	-99	-75	-128	-100

Table 4.12 Brightness pre-set values as configured for intensities 1-5

INTENSITY		**1**		**2**		**3**		**4**		**5**	
PROP	**TYPE**	**min**	**max**	**min**	**max**	**min**	**max**	**min**	**max**	**min**	**max**
Brightness	**I**	0	24	25	49	50	74	75	99	100	128

It was deemed that anything below -128 or above 128 introduced too significant of a reduction, this significance can already be seen in intensities 4 and 5 of the dark augmentation in Figure 4.24 where a large number of pixels have become black, almost hiding the leaf pattern within these samples. While a completely black image is a possible occurrence, for example at night, it is not particularly useful for evaluation.

Sun flare. The sun flare augmentation has 6 configurable properties as described in Table 4.3. As with many of the following augmentations, the interaction between the value of properties becomes more difficult to define as the number of properties increase. This hinders the creation

of systematically defined values that increase in intensity. For this augmentation, the first four intensities follow a systematic procedure for their pre-set value ranges while intensity 5 has its own configuration. This is necessary as if the systematically produced value ranges from the first four intensities were extended to intensity five, the resultant image would have to strong of a sun flare applied. The property ranges used for each intensity of this pre-set are provided in Table 4.13 and were used to produce the intensity samples as shown in Figure 4.25.

Intensity 1 Intensity 2 Intensity 3 Intensity 4 Intensity 5

Figure 4.25 Sample of the 5 intensities of the sun flare pre-set applied to Figure 4.22

Table 4.13 Sun flare pre-set values as configured for intensities 1-5

INTENSITY		**1**		**2**		**3**		**4**		**5**	
PROP	**TYPE**	**min**	**max**	**min**	**max**	**min**	**max**	**min**	**max**	**min**	**max**
Temperature	**I**	5000	7000	5000	7000	5000	7000	5000	7000	5000	7000
PROP	**TYPE**	**min**	**max**	**min**	**max**	**min**	**max**	**min**	**max**	**value**	
Rounds	**I**	5	10	10	15	15	20	20	25	30	
Size	**I**	41	51	38	46	36	42	35	39	107	
PROP	**TYPE**	**value**		**value**		**value**		**value**		**value**	
Intensity	**I**	12		16		22		30		85	
MaxPoints	**I**	50		75		100		125		80	
MinPoints	**I**	1		26		51		76		40	

For the first four intensities, the *Intensity*, *MaxPoints*, and *MinPoints* properties are all fixed values that increase along with intensity. *MaxPoints* and *MinPoints* increase by a constant value of 25 on each step whereas *Intensity* starts at 12 cumulatively grows such that the value of intensity x can be modelled as $Intensity = 12 + 2x$. *Rounds* and *Size* are variable properties with a maximum and minim randomisation window. The maximum and minimum values for *Rounds* are incremented by 5 for each intensity increment. The procedure used to determine *Size* is more complex as each intensity is determined based on the maximum and minimum values of the previous steps. Using values A, B, and i where A is the maximum value for an

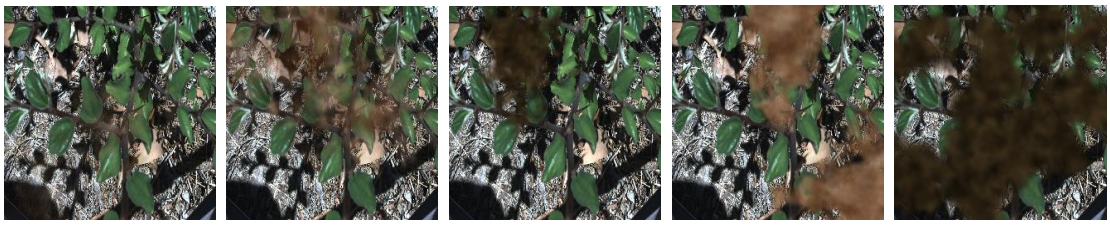
intensity, B is the minimum value for an intensity, $A_1 = 51$, $B_1 = 41$, and i is the intensity pre-set level, the following equations can be used to determine the maximum and minimum values of *Size* for the first for intensities.

$$A_i = A_{i-1} - (A_{i-1} - B_{i-1})$$

$$B_i = A_i - \frac{(A_{i-1} - B_{i-1})}{2}$$

For the fifth intensity, *Intensity*, *MaxPoints*, *MinPoints*, *Rounds*, and *Size* are all fixed values determined empirically and are unrelated to the previous four intensities. All 5 intensities share the same values for *Temperature* which is configured as a variable property with a minimum of 5,000 and a maximum of 7,000. This range was derived from the approximate colour of atmospheric sunlight, 5,900K [101], by generously rounding 5,900K to 6,000K and allowing variance of 1,000K either side. This range allows for atmospheric lighting temperature changes that occur during the day providing a colour range of warm-yellow to cool-white.

Dirt. The dirt augmentation has the largest number of configurable properties of any of the augmentations with 12 properties listed in Table 4.4. As observed with the sun flare augmentation, this further obscures the interactions between each of the properties making procedurally determined differences between intensities appear much more arbitrary. In addition, the dirt augmentation cannot be easily increased in intensity through systematic methods due to the complexity of the interactions between configurable properties. To work around this, the pre-sets for the first augmentation have been created in two groups, where intensities 1 and 2 create light and strong dust and intensities 3 – 5 create light, medium, and heavy mud. The property values for these pre-sets are provided in Table 4.14, and were used to generate the images shown in Figure 4.26



Intensity 1 Intensity 2 Intensity 3 Intensity 4 Intensity 5

Figure 4.26 Sample of the 5 intensities created for the dirt augmentation applied to Figure 4.22

Table 4.14 Dirt pre-set values as configured for intensities 1-5

INTENSITY		**1**		**2**		**3**		**4**		**5**	
PROP	**TYPE**	**min**	**max**	**value**		**min**	**max**	**min**	**max**	**min**	**max**
Scale	**D**	1	1.5	1		1.1	1.5	1	1.25	0.9	1
PROP	**TYPE**	**min**	**max**	**min**	**max**	**value**		**min**	**max**	**min**	**max**
Opacity	**I**	32	64	64	96	64		64	72	64	96
PROP	**TYPE**	**min**	**max**	**min**	**max**	**value**		**value**		**value**	
DirtPatches	**I**	1	2	2	4	1		2		4	
Particles	**I**	1000	2000	1000	2000	1000		750		500	
PROP	**TYPE**	**value**		**value**		**value**		**value**		**value**	
ClumpSize	**I**	40		40		40		40		40	
ColourPalette	**DIRTFX::COLOUR PALETTE**	Palette1; Palette2; Palette3		Palette1; Palette2; Palette3		Palette1; Palette2; Palette3		Palette1; Palette2; Palette3		Palette1; Palette2; Palette3	
DoNot ReusePoints	**B**	TRUE		TRUE		FALSE		FALSE		FALSE	
MaxDistance	**I**	400		400		400		400		400	
MaxRounds	**I**	200		200		200		200		100	
MinPoints	**I**	6		6		6		6		6	
SmoothingAttempts	**I**	12		12		12		12		12	
SmoothingIntensity	**I**	3		3		3		3		3	

Although configurable and can be used to make noticeable alterations to generated augmentations, the properties *ClumpSize*, *ColourPalette*, *MaxDistance*, *MaxRounds*, *MinPoints, SmoothingAttempts and SmoothingIntensity* are consistent in each pre-set. With the exception of *ColourPalette*, these properties were used for both augmentation size and intensity

scaling. These properties were largely superseded by the later-added *Scale* property reducing the need for their configuration. The *ColourPalette* is purely a decorative property with no significant variation required between intensities.

The most notable difference between the two pre-set groups is the value assigned to the property *DoNotReusePoints*. As described in Section 4.1.4, when 'True' this property limits the overall complexity of applied augmentations by only allowing points to be used once when creating polygons. This property is 'True' for pre-sets 1 and 2, and false for pre-sets 3, 4, and 5.

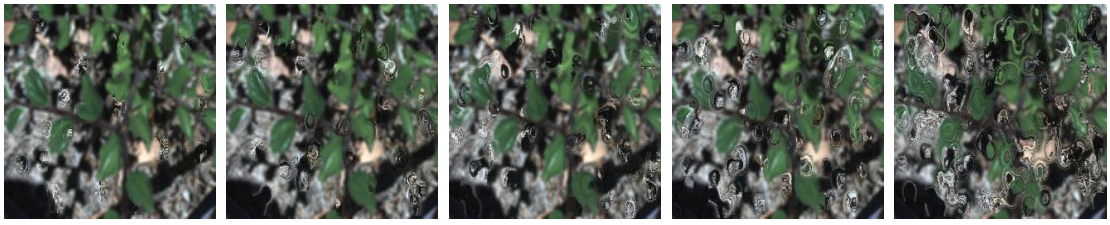
For pre-sets 1 and 2, the property *Particles* is consistently configured to use a random number between 1000 and 2000. This value is higher than those seen in pre-sets 3, 4, or 5 for the *Particles* property and is related to the *DoNotReusePoints* property in that if point reuse is denied, more points are required to ensure sufficient polygon creation. For pre-sets 3, 4, and 5, the value of *Particles* starts at 1000 and is reduced by 250 at each pre-set. This decrease is intended to offset the increase in *DirtPatches* as the total number of points within each dirt augmentation can be expressed as $DirtPatches \times Points$. For pre-sets 3, 4 and 5, this makes the total number of points per augmentation 1000, 1500, and 2000 respectively. Rather than increase the number of points, using multiple dirt patches generates new cluster of *Points* at a new location within the image creating larger dirt patches where dirt patches generate close together. While this logic holds true for pre-sets 1 and 2 also, the need to increase *DirtPatches* is to compliment *Particles* rather than balance it by generating new dirt patches that either accentuate existing patches or increase the dirt coverage.

Opacity determines the transparency of each drawn polygon, for pre-sets 1 and 2, the opacity is selected from a 32-value window starting from 32 and 64 respectively. Pre-sets 3, 4, and 5 on the other hand do not have a shifting window, but rather a growing window which starts at a fixed value of 64 which becomes a variable value over the remaining two pre-sets by incrementing the max-value by 16 for each intensity level. This approach was used instead of the sliding window to ensure the dirt generated by pre-set 3 maintained a valid jump in intensity from pre-set 2, but also to allow the augmentation to appear "thicker" for pre-sets 4 and 5.

The final property is *Scale*. The value of *Scale* is not used when fitting the generated augmentation to the image, rather it is used to calculate the size of the canvas the augmentation is generated on. As a result, increasing the value of *Scale* decreases the size of the generated augmentation The final augmentation is always resized to the size of the image before it is

applied. For pre-sets 1, the scale starts with a variable window of 1 – 1.5 which is decreased to a fixed value of 1 for pre-set 2. This was intended to ensure that pre-set 2 appears larger than pre-set 1, while allowing for a degree of randomisation on pre-set 1. For pre-sets 3, 4, and 5, each pre-set increment has a smaller variable window with smaller maximum and minimum values creating larger dirt patches. At each pre-set, the minimum is decreased by 0.1 and the maximum by 0.25.

Focused Rain. The focused rain augmentation has five configurable properties which are listed in Table 4.6. Unlike the other augmentations, no standardised pre-set was created for this augmentation as even with effort to make this augmentation scale based on the dimensions of the input image, dataset-specific augmentations yielded more believable results. The values used to dataset-specific pre-sets are discussed in the following sections. Table 4.15 lists the property values used for each intensity to generate the images as seen in Figure 4.27. Appendix H-1 provides the alternate property values.



Intensity 1 Intensity 2 Intensity 3 Intensity 4 Intensity 5

Figure 4.27 Sample of the 5 intensities created for the focused rain augmentation generated using the DeepWeeds pre-set

Table 4.15 Focused rain pre-set values as configured for intensities 1-5 (DeepWeeds version)

INTENSITY		**1**		**2**		**3**		**4**		**5**	
PROP	**TYPE**	**min**	**max**	**min**	**max**	**min**	**max**	**min**	**max**	**min**	**max**
Drops	**I**	25	50	50	75	75	100	100	150	200	400
PROP	**TYPE**	**value**		**value**		**value**		**value**		**value**	
Background Blur	**I**	5		5		5		5		5	
Foreground Blur	**I**	1		1		1		1		1	
Refraction	**I**	20		20		20		20		20	
Tolerance	**I**	10		10		10		10		10	

The properties *ForegroundBlur*, *Refraction*, and *Tolerance* are not changed across any of the intensities across any of the three pre-sets. It was found that they had a negligible effect better

controlled through *Drops*. *Drops* follows a pattern for the Wellington Camera Traps and Open Images V6 datasets starting with a minimum value of 25 and a maximum of 50 at intensity one which increases by setting each new intensities minimum to the previous of the last and doubling the range. It was not possible to create a more procedural increase for the DeepWeeds pre-set and as a result, it has ranges are as specified in Table 4.15. The *BackgroundBlur* property is also unique across each pre-set and must be increased to compensate for an increase in image size. Naturally, as the DeepWeeds dataset has the smallest images, it has the smallest *BackgroundBlur* value.

Unfocused Rain. The unfocused rain augmentation has six configurable properties which can be found in Table 4.6 and Table 4.7. Of these properties, only one is changed across each pre-set. Two sets of pre-sets were created for this augmentation: The standardised pre-set which was used by the Wellington Camera Traps dataset and the Open Images V6 Trees dataset and a slightly less-intense, DeepWeeds pre-set used for the DeepWeeds dataset. Separate pre-sets were required for the DeepWeeds dataset as the scaling built into the unfocused rain augmentation was insufficient to create augmentations of similar intensities consistently across each of the datasets. Using the standardised pre-set on the DeepWeeds dataset resulted in overly strong intensities. Creation of specialised pre-set was favoured due to insufficient time available for re-programming. The property values used to generate the images in Figure 4.28 can be found in Table 4.16. Configuration for the Wellington Camera Traps and Open Images V6 Trees dataset can be found in Appendix H-2.

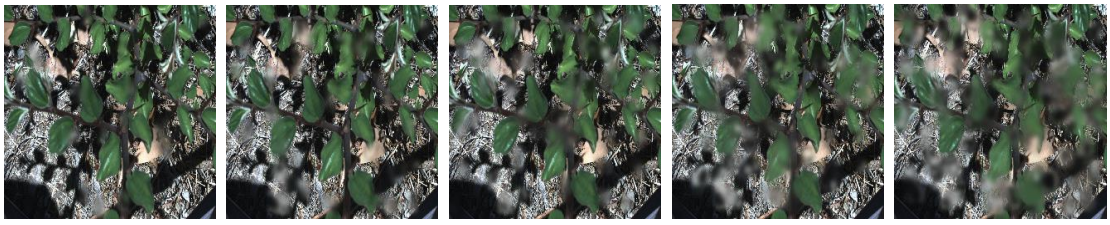


Figure 4.28 Sample of the 5 intensities created for the unfocused rain augmentation applied to Figure 4.22

Table 4.16 Unfocused rain pre-set values as configured for intensities 1-5

INTENSITY		1		2		3		4		5	
PROP	**TYPE**	**min**	**max**	**min**	**max**	**min**	**max**	**min**	**max**	**min**	**max**
Drops	**I**	25	50	50	75	75	100	100	125	125	150
Intensity	**I**	32	64	32	64	32	64	32	64	32	64
PROP	**TYPE**	**value**		**value**		**value**		**value**		**value**	
ForegroundBlur	**I**	1		1		1		1		1	
Refraction	**I**	20		20		20		20		20	
Tolerance	**I**	10		10		10		10		10	

As the nature of the unfocused rain augmentation requires the background object to be in focus, a change in intensity of any image in Figure 4.28 to one of its direct neighbours can be hard to notice. Despite this difficulty, the number of drops is configured to increase for each intensity. For both pre-sets, a variable property is used which allows a randomised number of drops for each generated augmentation. With each jump in intensity, the minimum value for this range is set to the maximum value for the previous intensities' range. For the standardised pre-set, the size of this range doubles that of the previous range whereas the DeepWeeds pre-set maintains a constant 25 drop range size across each intensity. By not doubling the range of *Drops* for each intensity in the DeepWeeds pre-set, a slightly less intense version of the unfocused rain augmentation can be achieved which looks visually similar to the standard pre-set intensities applied to the non-DeepWeeds datasets. Figure 4.29 shows intensity 5 as configured by the standard pre-set, which is intended to be of a visually intensity when compared to Figure 4.28.

Figure 4.29 Example of intensity 5 unfocused rain augmentation configured using the standard pre-set

Frost. The frost augmentation has no unique configurable properties, inheriting only the three configurable properties listed in Table 4.5 that are common across the four water-based augmentations. Configuring pre-sets for the frost augmentation is difficult as the augmentation looks less realistic as the value of *Drops* decreases. This can be partially resolved by reducing the *Refraction* value as *Drops* is decreased which makes the refraction applied less noticeable. However, to achieve the full frost augmentation, a large number of drops is needed to ensure image coverage which has the side effect of creating a very strong augmentation. To ensure augmented images were not completely blurred out, small *Refraction* values were used in the created frost pre-set. Figure 4.30 contains five examples of different intensities created using the pre-set values provided in Table 4.17.

Intensity 1 Intensity 2 Intensity 3 Intensity 4 Intensity 5

Figure 4.30 Sample of the 5 intensities created for the frost augmentation applied to Figure 4.22

Table 4.17 Frost rain pre-set values as configured for intensities 1-5

INTENSITY		**1**	**2**	**3**	**4**	**5**
PROP	**TYPE**	**value**	**value**	**value**	**value**	**value**
Drops	**I**	200	400	600	800	1000
Refraction	**I**	1	2	3	4	5
Tolerance	**I**	10	10	10	10	10

The properties of *Drops* and *Refraction* start at 200 and 1 respectively for intensity 1 and are increased by consistently by 200 and 1 for each subsequent intensity. The increase in *Drops* aims to increase the coverage of the applied frost as the intensity increases, more noticeable on larger images, while the increase in *Refraction* aims to increase the frost thickness through stronger refraction. The value of *Tolerance* is set at 10 for each intensity as there is no notable benefit of altering this value.

Water Residue. The water residue augmentation like the frost augmentation also has not unique configurable properties, inheriting only the three configurable properties listed in Table 4.5. The water augmentation makes use of a significant number of translucent white drops which can easily look unnatural depending on the configuration used. To mitigate this, it is recommended to favour lower values (fewer than 500) for *Drops* and increasing both *Refraction* and *Tolerance* when increasing intensity. Furthermore, using too high of a value for *Drops* will result in long augmentation generation times making it difficult to utilise *Drops* for intensity scaling. Figure 4.31 contains five examples of different intensities created by using the pre-set values provided in Table 4.18.

Intensity 1 Intensity 2 Intensity 3 Intensity 4 Intensity 5

Figure 4.31 Sample of the 5 intensities created for the water residue augmentation applied to Figure 4.22

Table 4.18 Water residue rain pre-set values as configured for intensities 1-5

INTENSITY		**1**		**2**		**3**		**4**		**5**	
PROP	**TYPE**	**min**	**max**	**min**	**max**	**min**	**max**	**min**	**max**	**min**	**max**
Drops	**I**	50	100	100	150	150	200	200	250	250	300
PROP	**TYPE**	**value**		**value**		**value**		**value**		**value**	
Refraction	**I**	0		10		20		30		40	
Tolerance	**I**	50		60		70		80		90	

Although it is hard to see a severity increase in Figure 4.31, each image from left to right contains an increasing number of translucent white watermarks. While these intensities do not appear to create significant augmentations when compared with those of the frost or dirt augmentations, it was identified in Chapter 2.2 that IR and OD models can be thrown off with even minor image modifications. Furthermore, this augmentation recreates what is naturally a very minor augmentation, and as such is difficult to strengthen to the same level as other augmentations.

All properties in the water residue pre-set are incremented with each intensity. *Drops* is configured as a variable property that for each intensity has a range of 50 starting from 50 and increases by 50 for each successive intensity. *Refraction* and *Tolerance* start at 0 and 50 respectively for intensity 1 and increase by 10 with each successive intensity.

The pre-sets mentioned in this section are available alongside the generalised DeepWeeds training program on request to the author.

4.2.2. Dataset Creation

The bulk generation mode provided by the artificial augmentation generation program supports the creation of multiple augmented datasets during one session. This is achieved by creating a new augmented dataset in its own folder for every intensity in the provided pre-set file.

The bulk generation program must be run individually for each dataset as it only supports a single input source. The end result of this process is 45 augmented datasets (9 augmentations $\times$ 5 intensities) named by their augmentation and intensity.

In total, 135 augmented datasets were created for this thesis which are available alongside the generalised DeepWeeds training program on request to the author.

4.3. Summary

In this section, the concept of configurable properties, which is a setting that alters an augmentation's generation through either a fixed value or a randomised value, has been introduced. Each augmentation has at least one of these properties which can be configured either through the generation tool's user interface or by creating pre-set files for the bulk generation utility.

For this thesis, 135 unique datasets were created using the pre-set configurations listed in Section 4.2.1. These datasets are used in the next chapter for evaluation of the three unaugmented datasets.

5. Results and Discussion

This chapter discusses the use of artificially generated augmentations produced as described in Chapter 4 to generate artificially modified datasets, for the purposes of creating additional training data to improve robustness in deep learning models.

In this project, nine artificially reproduced augmentations have been designed to replicate their respective real-world appearances. This task requires precisely replicating real-world natural phenomena which is challenging, however, if correctly done should allow these augmentations to be considered as substitutions for natural augmentations. This substitution should remove the need for natural augmentations to be obtained, significantly optimising the process of robustness training.

To validate utilisation of the artificially generated augmentations generated in this paper in model robustness, this chapter explores the following questions in the context of each augmentation:

- *"Does the artificial augmentation impact the performance of a non-robust model?"*
- *"Does the artificial augmentation improve performance of a non-robust model?"*
- *"Does the artificial augmentation replicate the performance of a natural augmentation?"*

Impact. This question, while simple, explores whether the generated augmentations actually impact the performance of a deep learning model as would be expected from a natural augmentation. This is a particularly important question as the generation of semi-random augmentations depends on pre-set values which are ultimately configured to resemble what a person would see and not what a model sees. Section 5.1 explores this question by comparing model performances on each intensity of each augmentation to the baseline metrics established in Chapter 3.

Improve. The second question explores how much, if at all, the artificially augmented datasets would improve an existing model when used as a supplemental training dataset. To fully address this question would require the creation of an additional 135 models (5 cross-fold models x 3 datasets x 9 augmentations) with intensities being ignored. This was not an achievable goal within the scope of project. However, Section 5.2 explores a subset of this question by reducing the scope of analysis to the best cross-fold model from the DeepWeeds

and Wellington Camera Datasets on four of the augmentations, providing an indication on whether or not these augmentations improve a non-robust model.

Replicate. The last question explores the overall outcome of this project, whether or not the artificially generated augmentations mimic the impact of real-world augmentations. This question requires a comparison between two models where one model has been trained on a dataset containing naturally occurring augmentations and the other a dataset containing artificially generated augmentations. To fully perform this validation for the augmentations in this project, each of the augmented datasets requires a similar sized and balanced counterpart dataset where each image has both an unaugmented and naturally augmented copy. This is not possible in the context of this project as the three unaugmented datasets used to generate each of the augmented datasets have been obtained from various online resources sourced from a variety of geographical locations and rely on unreproducible events, such as animal positioning or environmental factors. To partly address this important question, Section 5.3 provides an object detection case study which contains some of the augmentations this paper provides. This case study contains near-identical unaugmented and augmented image variants for some images enabling creation of a minimal dataset to provide an indication of the answer to this question.

Finally, Section 5.4 discusses the answers to these questions.

5.1. Augmentation Impact Evaluation

Each augmented dataset created for this project used a specifically designed set of pre-sets, as discussed in Section 4.2, that enabled the creation of five levels of procedurally stronger intensities for each augmentation. In total, 135 augmented datasets were made from the starting three unaugmented datasets introduced in Chapter 3. Each of the artificial augmentations alter images in different ways, such as through occlusion or distortion, making it difficult to balance the intensities of each pre-set such that an intensity five focused rain augmentation is equivalent to an intensity five dirt augmentation. Although each pre-set was created with this problem in mind, the pre-sets are configured such that they appear visually balanced to the human eye rather than what might be considered balanced from a machine learning perspective. This is further complicated, as discussed in Section 2.2, by the fact that it is very difficult to tell how

a model perceives image features, making it difficult to optimise either pre-sets or artificial augmentations for a particular model.

This section identifies the impact on performance when evaluating a model on artificially generated augmented data. Although it might be hypothesised that model performance decreases proportionally with any augmentation (either artificial or natural) as augmentation intensity increases, this hypothesis requires verification.

For the image recognition datasets, the evaluation results were determined in the same manner as the baseline metric, by using the generic DeepWeeds training program introduced in 3.2. For both datasets, their respective augmented datasets were evaluated using each of their respective five models produced in Chapter 3. The result of each evaluation was aggregated by averaging the five F1-Scores produced by each cross-fold model for each intensity of each augmentation. The result of this aggregation is a nine by five table of F1-Scores.

The augmented variants of the Open Images Trees dataset were also evaluated in the same manner as the baseline model in Section 3.5.2, producing a nine by five table of mAP scores where each cell represents the average mAP achieved by each cross-fold model for each intensity of each augmentation.

DeepWeeds.

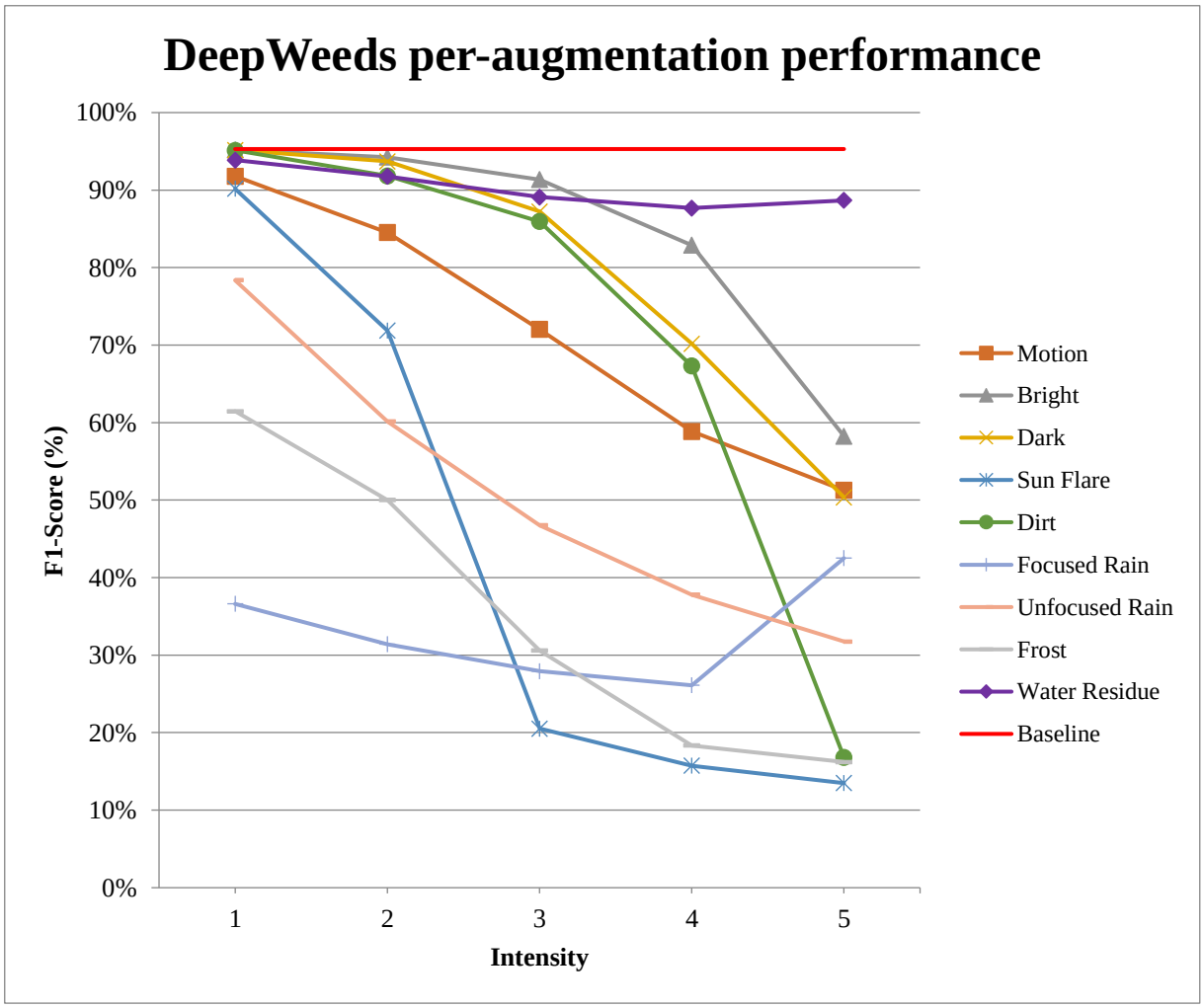


Figure 5.1 Graph of DeepWeeds performance over increasing intensities of artificial augmentations

Table 5.1 DeepWeeds model performance (F1-Score, %) over increasing intensities of artificial augmentations

DeepWeeds	***Intensity***				
Augmentation	**1**	**2**	**3**	**4**	**5**
Motion	91.8	84.5	72.0	58.9	51.3
Bright	95.2	94.3	91.4	82.9	58.3
Dark	95.2	93.7	87.2	70.2	50.4
Sun Flare	90.2	71.9	20.5	15.7	13.5
Dirt	95.1	91.8	86.0	67.3	16.8
Focused Rain	36.6	31.4	27.9	26.1	42.5
Unfocused Rain	78.4	60.1	46.8	37.8	31.8
Frost	61.5	50.0	30.6	18.3	16.2
Water Residue	93.9	91.8	89.1	87.7	88.7
Baseline	95.3				

Table 5.1 provides the F1-Score for every augmented dataset averaged over each of the five DeepWeeds cross-validation models. The per-fold values used to calculate each average are provided in Appendix I-1. The averaged values in Table 5.1 have been used to create the graph provided in Figure 5.1.

Figure 5.1 presents a set of interesting trends which with the exception of the water residue and focused rain augmentations, decrease in F1-Score as the intensity increases as hypothesised. Looking at the focused rain augmentation, for the first four intensities, the performance per augmentation is low, but confirms with trend observed in the other augmentations. At intensity five, there is a steep increase in performance which results in a better F1-Score than what was reported for intensity one. As the per-fold F1-Score metrics reported in Appendix I-1 show that this increase is consistent across each fold, it is likely that the increase in performance is caused by an issue with the augmented dataset.

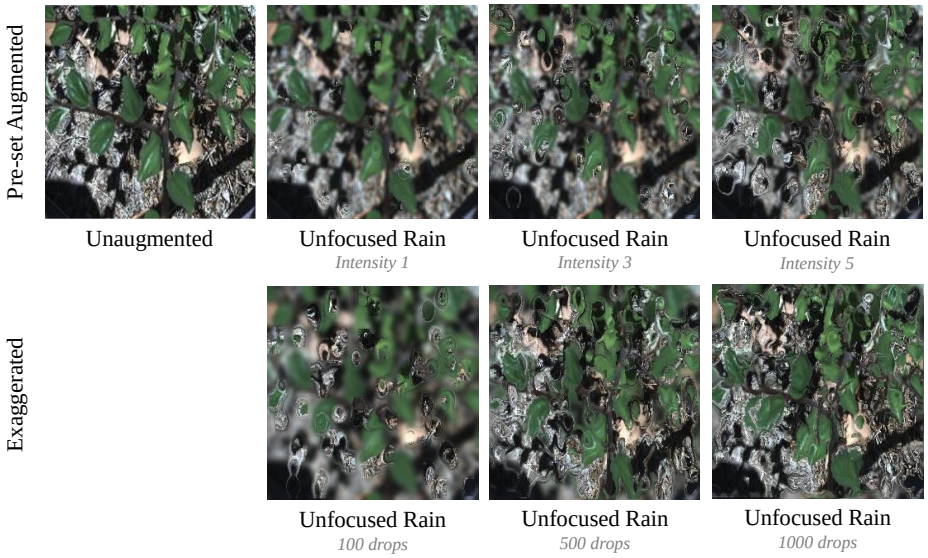


Figure 5.2 Comparison showing the impact on increasing drop count for Focused Rain augmentation. Unaugmented image Adapted from [100] (Apache 2.0 License)

Figure 5.2 contains two rows of images. In the top row, three augmented versions of the upper-left image generated at different intensities have been taken from the DeepWeeds focused rain augmented dataset. Underneath each augmented image is a relatively stronger augmentation created by increasing the number of drops on the image to exaggerate its respective intensity.

In addition, each of the stronger augmentations has a background blur value of 25 instead of 5 to better separate blurred pixels to refracted pixels.

The strong augmentations in Figure 5.2 help to show the impact of adding additional raindrops to the image. Compared to the image with 100 drops, the 1000 drop image appears to be much less distorted despite having a 'stronger' augmentation. Notably, even with the majority of the blurred pixels in the image with 100 drops being replaced by refracted raindrops, the image with 1000 drops appears to be closer in resemblance to the initial unaugmented input image. The same difference can be seen between Intensities one and five in the top row, although this effect is not as easily noticeable in these images. This issue is caused by a combination of the small image size, which effectively merges drops as a side effect of limited placement area and a small error in this augmentation's refraction code which incorrectly flips the refraction image when replacing the normal texture. These issues are minor independently, however the combination of large refraction areas as a result of closely spaced drops amplifies the issue during refraction. This has been resolved for future usage of the augmentation by updating the refraction code used by this augmentation. As the other datasets have larger images, the impact of this issue is significantly reduced. Based on this discussion, the trend observed in the focused rain augmentation can be attributed to an anomaly within the dataset. This is a candidate for future work which can recreate the augmented dataset using the corrected code to confirm the significance of this issue.

The other augmentation to not meet the hypothesis is the water residue augmentation. This augmentation follows a similar trend to the focused rain augmentation in that the performance increases after the fourth intensity, however by a less significant amount. Once again, the F1-scores reported for intensity 5 and provided in Appendix I-1 are consistent with each other, indicating this is most likely an issue with the augmented dataset. Referring back to Figure 4.31 (page 98), there do not appear to be any inconsistencies or anomalies present on any of the images that would suggest an issue with the augmentation, suggesting instead the issue lies with the pre-set configurations used. In Section 4.2.1, it was observed that the difference between intensities of the water residue augmentation were difficult to notice. This observation was attributed to the nature of the augmentation, as watermarks are largely unobtrusive. However, as proposed by the discussion point from Section 2.2 on human versus deep learning perception, the minor differences could still impact a machine learning model's predictions. As the first four intensities show progressive performance loss, it would appear that the augmentation is discernible by the model. Overall, while this augmentation may require a

stronger set of pre-sets, this trend does not appear to signify an issue with the augmentation itself.

The sun flare augmentation produces a large drop in performance after intensity two suggesting that the sun flare augmentation has significantly increased in intensity at intensity three. Figure 4.25 does not appear to show a significant change between intensities two and three. In fact, arguably a stronger change can be seen between intensities three and four, however, this is not reflected in the graph.

Figure 5.3 Comparison of four sun flare: intensity 2 images to four sun flare: intensity 3 images

Figure 5.3 compares four images augmented with the sun flare augmentation configured at intensity two to the same four images configured at intensity three. From a visual inspection, the images at intensity two appear consistent with the portrayal of intensity two in Figure 4.25, however, the images at intensity three appear slightly stronger. This can be easily attributed to chance as the sun flare augmentation uses randomisation at each of its intensities. Another noticeable difference between the two intensities is the appearance of localised areas of higher brightness. All the images configured at intensity two appear to contain a consistent re-colour applied with very subtle brightness changes, whereas the images configured at intensity three contain much more intense differences in brightness. This could signify that the model is impacted by the uneven brightness.

Comparing the sun flare augmentations in Figure 4.25 to the brightness augmentations in Figure 4.24, the sun flare augmentation at intensities one and two appear mostly visually

consistent with the brightness augmentation at intensities two and three, with the only difference being temperature adjustment and a slight unevenly applied brightness. Despite the similarity, the sun flare augmentation at intensity one has a lower F1-Score than any of the first three intensities of the brightness augmentation, suggesting that colour temperature has a bigger impact on the DeepWeeds models predictions than a brightness adjustment alone. Furthermore, intensities four and five of the sun flare augmentation appear to increase brightness in some areas beyond what is seen in the intensity five brightness augmentation. Following the trend of the Brightness augmentation, each progressive intensity appears to have a greater impact on the performance than the previous, suggesting that the DeepWeeds models are less tolerant to brightness adjustments the more the brightness increases. Based on this, it is likely that the combination of increasing brightness and temperature adjustments is responsible for the steep drop in performance between intensities two and three.

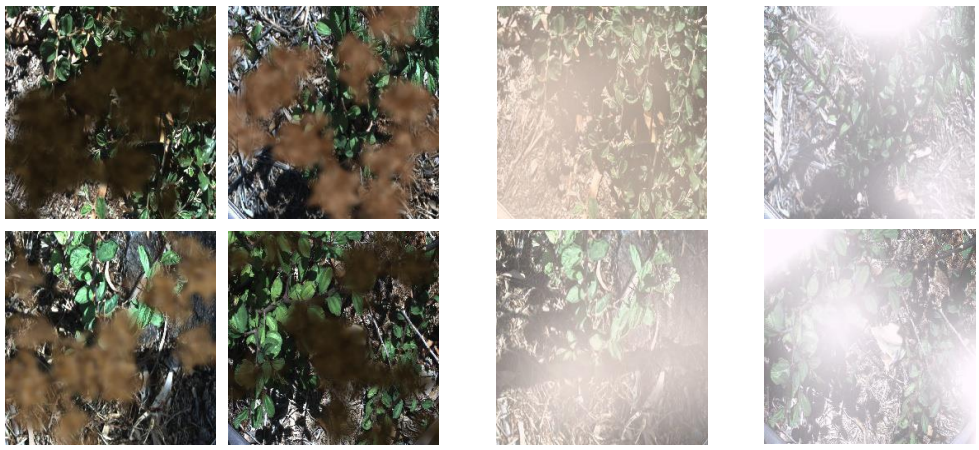


Figure 5.4 Comparison of four dirt: intensity 5 images to two sun flare: intensity 4 and two sun flare: intensity 5 images

It is also interesting to note that the performance values for intensities three through five of the sun flare augmentation are very close to the performance value for intensity five dirt. The overall trend for the dirt augmentation is easily attributable to incremental introduction of stronger occlusion. At intensity five, the dirt augmentation adds multiple dirt patches to the image, occupying a large area of image and hiding recognisable image features. Figure 5.4 provides a comparison between intensity five of the dirt augmentation and intensities four and five of the sun flare augmentation. From this comparison, it can be seen that the sun flare

augmentation does not typically create as much occlusion, and where it does, it is usually translucent.

The remaining four augmentations (frost, unfocused rain, motion blur, and darkness) are all relatively similar in terms of performance loss per intensity increase. The most notable difference between these augmentations are their performance values for intensity one, which can be easily explained by the level of augmentation they offer. This does mean that model does not see the pre-sets values for each of these augmentations at intensity one as consistent, but this is largely unavoidable due to augmentations such as frost being far harder to apply at visually low intensities.

Overall, the results presented in Table 5.1 show that with the exception of the water residue and focused rain augmentations, the augmentations followed the expected trend of reducing model performance as the intensity increases. Future work should include further exploration of the water residue and focused rain augmentations to better understand this. It has been observed that the DeepWeeds models appears to struggle with both large brightness adjustments and changes in colour temperature. This vulnerability presents an ideal test case for model training on the brightness and sun flare augmentations.

Wellington Camera Traps.

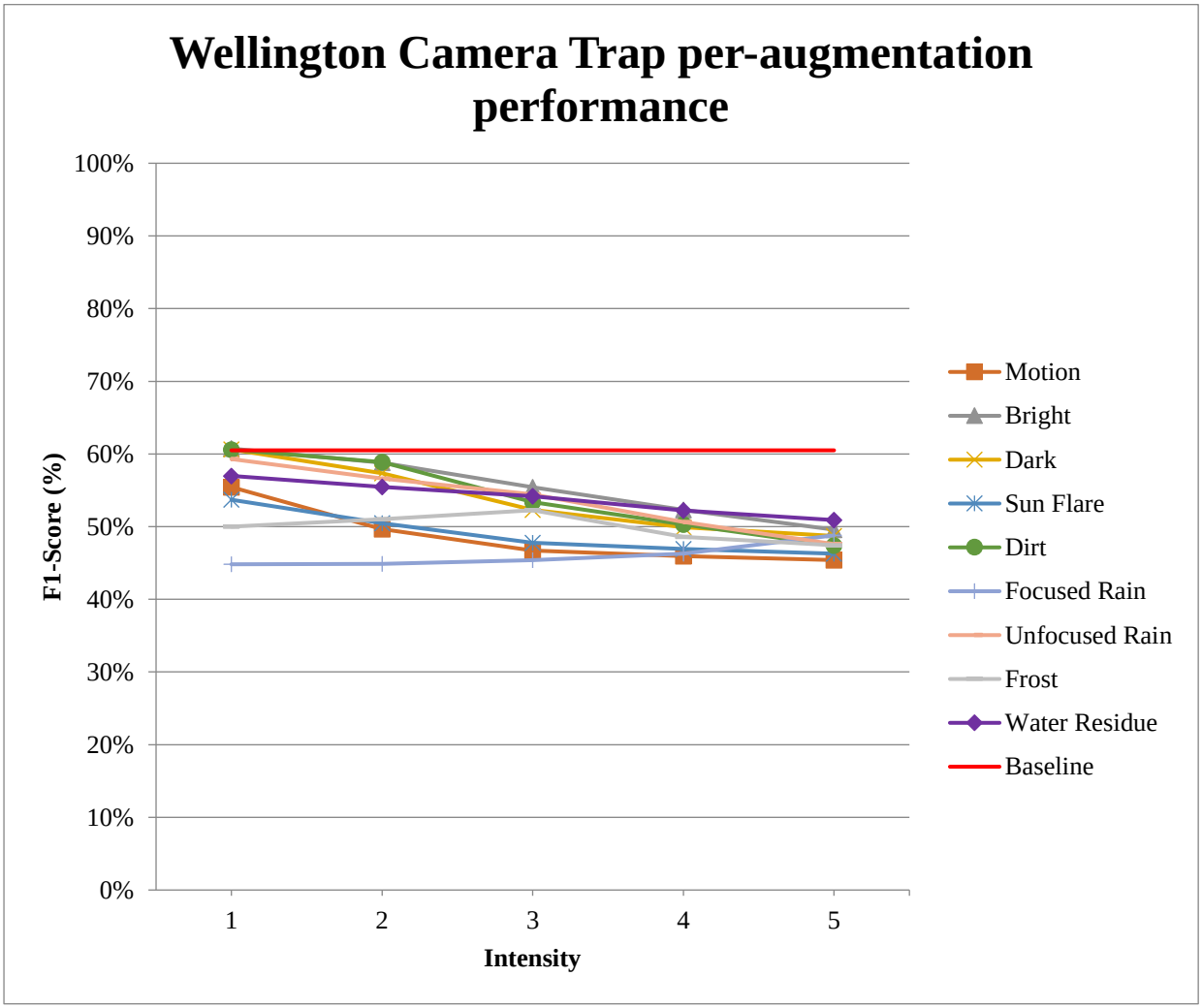


Figure 5.5 Graph of Wellington Camera Trap performance over increasing intensities of artificial augmentations

Table 5.2 Wellington Camera Traps model performance (F1-Score, %) over increasing intensities of artificial augmentations

Wellington Camera Traps	***Intensity***				
Augmentation	**1**	**2**	**3**	**4**	**5**
Motion	55.4	49.6	46.7	46.0	45.4
Bright	60.7	58.8	55.4	52.3	49.6
Dark	60.6	57.4	52.3	49.9	48.7
Sun Flare	53.7	50.5	47.8	46.9	46.3
Dirt	60.6	58.9	53.4	50.3	47.4
Focused Rain	44.8	44.9	45.4	46.3	48.8
Unfocused Rain	59.3	56.6	54.4	50.6	47.6
Frost	50.0	51.0	52.3	48.6	47.4
Water Residue	57.0	55.5	54.2	52.2	50.9
Baseline	60.5				

Table 5.2 provides the F1-Score for every augmented dataset averaged over each of the five DeepWeeds cross-validation models. The per-fold values used to calculate each average are provided in Appendix I-2. The averaged values in Table 5.2 have been used to create the graph provided in Figure 5.5.

The F1-Scores reported in Table 5.2 appear more clustered than the F1-Scores reported in Table 5.1. While this is difficult to justify from the numbers alone, this most likely stems from a combination of the lower baseline metric achieved in Chapter 3 and the smaller ratio of object to image bounds, addressed below, which effectively caps the augmented results.

The object to image size ratio, *Object-Image Ratio* (OIR), represents the scale of an object relative to the size of the image it is in. The smaller OIR was a desirable feature of the Wellington Camera Trap dataset as compared to the DeepWeeds dataset meaning objects do not occupy the whole image, instead they occupy a smaller area of the overall image. A smaller OIR means that there is more background, non-object, information in the image which naturally makes detection harder. In the context of augmentation, the smaller OIR means that there is a higher chance that pixels that represent image features will not be augmented and remain as they appear in the unaugmented version of the image.

Figure 5.6 Comparison of Wellington Camera Traps augmentation: Unaugmented, Dirt: Intensity 5 to DeepWeeds augmentation: Unaugmented, Dirt: Intensity 5. Wellington Camera Trap unaugmented image adapted from [91] (https://cdla.dev/permissive-1-0/). DeepWeeds unaugmented image adapted from [100] (Apache 2.0 License)

Figure 5.6 provides a comparison of two images, one from the Wellington Camera Traps dataset and the other the DeepWeeds dataset. Each image is presented as unaugmented and with an intensity five dirt augmentation. In this scenario, the object (Cat) of the Wellington Camera Trap dataset is partially obstructed by dirt, but a significant area of the unaugmented cat can still be seen within the augmented image. In contrast, the object (weed) of the DeepWeeds dataset is almost fully obstructed by the dirt augmentation. It is proposed that this

difference has helped the Wellington Camera Traps retain higher F1-Scores in comparison to the DeepWeeds dataset. This indicates the importance of considering OIR when designing pre-sets, another avenue for future work.

The focused rain and frost augmentations stand out as the only augmentations to have an increasing F1-Score as intensity increases. The frost augmentation does begin to decrease between intensities three and five, however, it is the reasoning behind the initial increase in performance which is of interest.

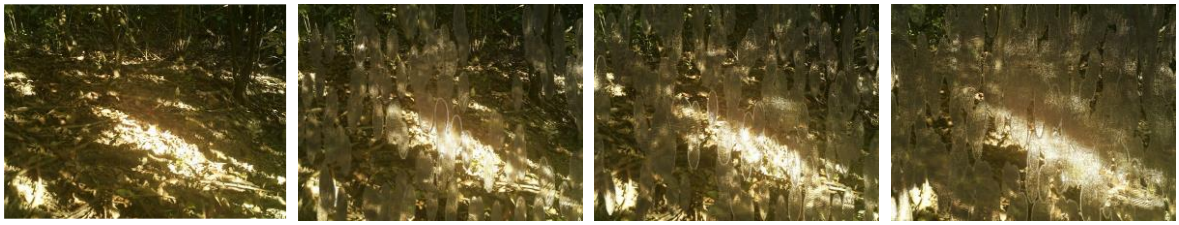


Figure 5.7 Comparison of Wellington Camera Traps augmentations: Unaugmented (left), Frost: Intensity 1 (centre, left); Intensity 2 (centre, right); Intensity 3 (right). Wellington Camera Trap unaugmented image adapted from [91] (https://cdla.dev/permissive-1-0/)

The performance increase over the first three intensities of the frost augmentation are in conflict with the observations made from the frost augmentation within the DeepWeeds performance results. Figure 5.7 compares the output of the first three intensities of the frost augmentation with each image shown appearing consistent, although scaled, with the first pre-sets as defined in Section 4.1.5.3. It is unclear why the stronger augmentation at intensity three resulted in a higher F1-Score than at intensity one, although a possible justification is that with a combination of the small OIR discussed earlier and random chance, the gaps present in the first three frost intensities allowed for good detection. However, in order to confirm this, further experimentation is required.

Figure 5.8 Comparison of Wellington Camera Traps augmentations: Unaugmented (left), Focused Rain: Intensity 1 (centre, left); Intensity 2 (centre, right); Intensity 3 (right)

The focused rain augmentation trend is likely the result of the same observation made in the DeepWeeds dataset in that the refraction is not to flipped correctly resulting in areas with large groups of raindrops which effectively remove the background blur. The effect of this is much less prominent within the results in comparison with the DeepWeeds dataset due to the larger size of the Wellington Camera Traps images.

The remaining augmentations appear to follow either one of two trends which ultimately converge around an F1-Score of ~48%. This is interesting as augmentations with similar trends also have similar y-axis values for each intensity, which is the opposite of what is seen in the DeepWeeds results where most augmentations may have a similar a trend, but also unique y-axis values. It is assumed that the smaller OIR of the dataset contributes to this observation as discussed earlier, but this could also indicate that the pre-sets are relatively balanced such that an intensity three dirt augmentation creates as much of a performance impact as an intensity three brightness augmentation. Another possible contributor to this trend is the fact that these results are reported based on a two-class model where one class performs significantly better than the other. This means that to remain fair to both classes, F1-Score adjustments are minimal and not spaced out well enough for a better analysis. For this dataset, analysing performance per class may yield more informative information on trends.

Overall, the results presented in Table 5.2 show a performance impact for each augmentation that, with the exception of the frost augmentation, are consistent with the earlier stated hypothesis that augmentation would reduce the model's performance as intensity increased.

Open Images V6 Trees

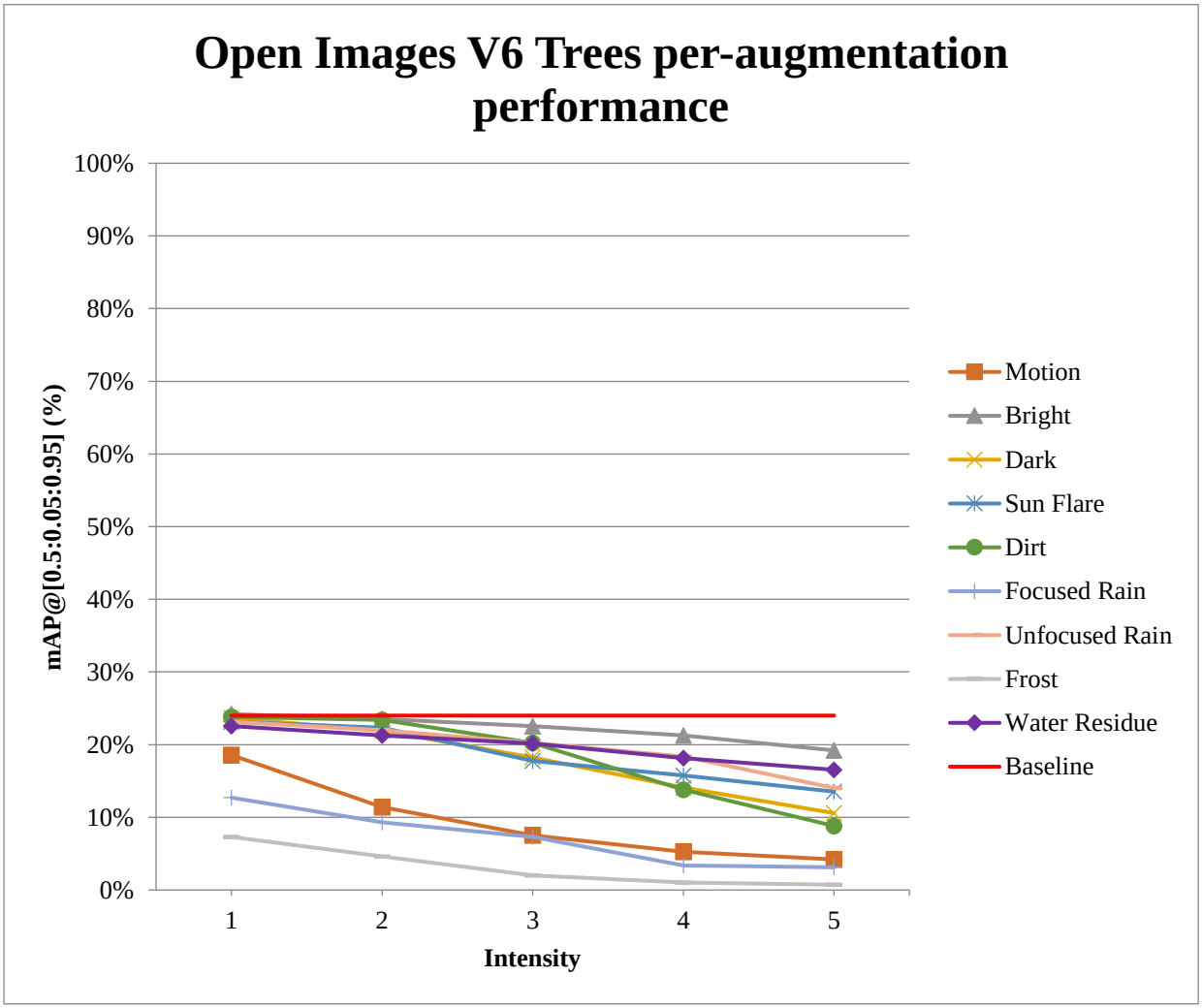


Figure 5.9 Graph of Open Images V6 Trees performance over increasing intensities of artificial augmentations

Table 5.3 Open Images V6 Trees model performance (mAP@[0.5:0.05:0.95], %) over increasing intensities of artificial augmentations

Open Images V6 Trees	***Intensity***				
Augmentation	**1**	**2**	**3**	**4**	**5**
Motion	18.6	11.4	7.5	5.3	4.2
Bright	24.2	23.5	22.6	21.3	19.2
Dark	23.7	21.9	18.2	14.1	10.6
Sun Flare	23.1	22.3	17.8	15.8	13.5
Dirt	23.8	23.4	20.2	13.8	8.8
Focused Rain	12.7	9.3	7.3	3.4	3.1
Unfocused Rain	23.1	22.0	20.2	18.4	14.0
Frost	7.3	4.6	2.0	1.0	0.7
Water Residue	22.6	21.3	20.2	18.1	16.5
Baseline	24.0				

Table 5.3 provides the F1-Score for every augmented dataset averaged over each of the five Open Images V6 Trees cross-validation models. The per-fold values used to calculate each average are provided in Appendix I-3. The averaged values in Table 5.3 have been used to create the graph provided in Figure 5.9.

The values reported Table 5.3 cannot be directly compared with the values reported in either Table 5.1 or Table 5.2 as they are reported using different metrics. However, the trends shown in Figure 5.9 appear to closely resemble those seen in the Wellington Camera Trap result graph shown in Figure 5.5 with the sun flare, frost and darkness augmentations being the three exceptions.

Unlike the previously discussed datasets, every augmentation reduces model performance as intensity increases. The consistency in both the performance drop per intensity and trends themselves were unexpected for this dataset, as this is the only dataset to contain images of non-uniform size, meaning that it was not feasible to check that the scaling of each augmentation worked well for every image. There are likely a number of reasons for this consistency, however, it most likely can be attributed to the combination of the underlying YOLO pre-trained model used and the YOLO framework itself.

Overall, the results presented in Table 5.3 show a performance impact for each augmentation which is consistent with the earlier stated hypothesis which predicted augmentation would reduce the model's performance as intensity increased.

5.2. Augmentation Training Evaluation

This section evaluates whether retraining a model on an augmented dataset improves its ability to deal with that augmentation. This is measured by comparing the resulting performance from a model not trained on augmented data (baseline model) to a model trained on augmented data (improved model).

To evaluate every model produced against every augmentation in this thesis is an exceptionally time-consuming task, which is outside of the scope for this thesis. Instead, this section takes the best performing cross-fold models from the DeepWeeds and Wellington Camera Dataset and trains them on the four augmentations which have naturally occurring versions. This results in a more achievable set of eight models which are discussed below.

Motion blur, brightness, darkness, and sun flare are the four augmentations evaluated in this section. For each of these augmentations, a new augmented dataset was created by randomly merging augmented images from the existing augmented datasets such that every image in the new dataset effectively had a random intensity. Using a mixed-intensity dataset was expected to better represent a real-world scenario where augmentations are unlikely to consistently appear at a single intensity.

Each improved model was trained using the generalised DeepWeeds training program introduced in Section 3.2 using its respective baseline model as a replacement for the pre-trained ResNet50 model. With the exception of only being trained over 64 epochs (to speed up training), all other parameters remained the same as specified in Chapter 3, including using the same cross fold splits ensuring that the test-data is unseen for both models.

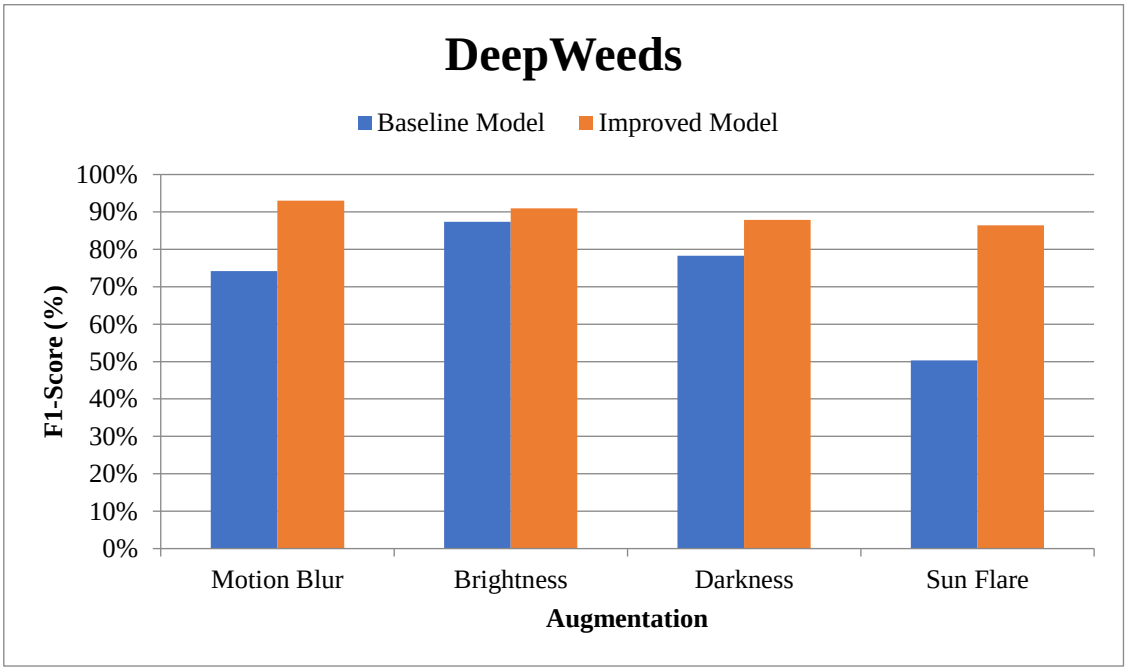


Figure 5.10 Graph showing performance between unaugmented and augmented trained models for the DeepWeeds dataset

The best performing DeepWeeds cross-fold baseline model was Fold 5 as shown in the training + validation set table in Appendix B. Figure 5.10 compares the performance of the baseline model and the improved model for each augmentation using the F1-Score percentage.

All four augmentations show an improvement between the baseline and improved models with the biggest improvement being seen in the sun flare augmentation. The improvements seen for

the sun flare and motion blur augmentations appears to indicate that the respective lower performance in the baseline model is indicative that the unaugmented DeepWeeds training data is insufficient to teach the baseline model to be robust against these augmentations. This is supported by the better baseline performance in the brightness and darkness augmentations which is both a minor part of the dataset, and one of the minor overfitting prevention augmentations utilised by the training program.

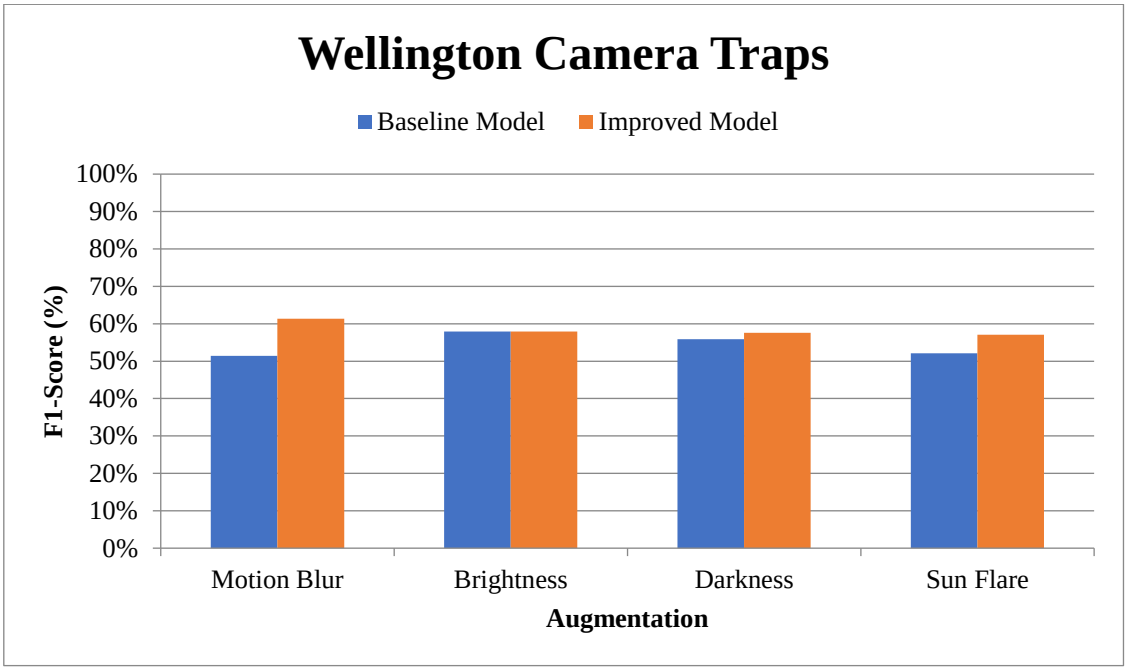


Figure 5.11 Graph showing performance between unaugmented and augmented trained models for the Wellington Camera Traps dataset

The best performing Wellington Camera Traps cross-fold baseline model was Fold 5 as shown in the training + validation set table in Appendix B. Figure 5.11 compares the performance of the baseline model and the improved model for each augmentation using the F1-Score percentage.

The performance improvements are smaller for this dataset than they were with the DeepWeeds dataset. All augmentations, with the exception of brightness, which is unchanged, show a minimal performance improvement.

The results shown in both graphs above indicate that in most cases, the motion blur, brightness, darkness, and sun flare augmentations generated in this thesis can be used to make models

more robust. The next step in this evaluation would be to expand this analysis to include the remaining five augmentations to see if there is any case in where this is not true.

A possible limitation of these results, however, is the 64-epoch training restriction. This restriction was used to expedite training and is potentially a contributor to the less significant improvement seen in the Wellington Camera Trap models. Another recommendation for future work would be to explore this further.

5.3. Bio-Heritage Project Case Study

This section evaluates the substitution of naturally occurring augmentations for artificially generated ones and compares model performance achieved between the two.

To compare natural augmentations, a dataset is needed that contains an unaugmented and naturally augmented copy of every image. There are very few existing datasets that meet this requirement, especially from outdoor environments, which makes acquisition of such a dataset very difficult. However, a small dataset was created using images acquired from the members of the University of Waikato Bio-Heritage project [102]. More information on this project can be found on their website [103].

The Bio-Heritage team captured a large collection of images from around the University of Waikato intended for tree identification. The images were acquired by mounting a camera to a car and driving around local roads at different times of the year, naturally creating duplicate images. As a result of the methodologies used to capture the images, a number of motion blur, brightness, darkness, and sun flare augmentations were present in the dataset, making it a good starting point for substitution.

In total, there were approximately 5,000 images to begin with, which was reduced to a small collection of 76 naturally augmented images which contained near identical unaugmented images. This is a substantial drop in image numbers compared to the other datasets, however, further illustrates the difficulty involved in acquiring natural augmentations. For each of the 76 images, a visually similar augmented image was created to replicate the natural augmentation, creating an augmented dataset. As the replicant images were only configured using the configurable properties provided for each augmentation, there were some minor differences between the two datasets.

For the comparison, each cross-fold model produced from establishing the Open Images V6 Trees dataset baseline in Section 3.5.2 was used to evaluate the performance of each dataset using the YOLO validation utility. The results of these evaluations can be seen in Figure 5.12.

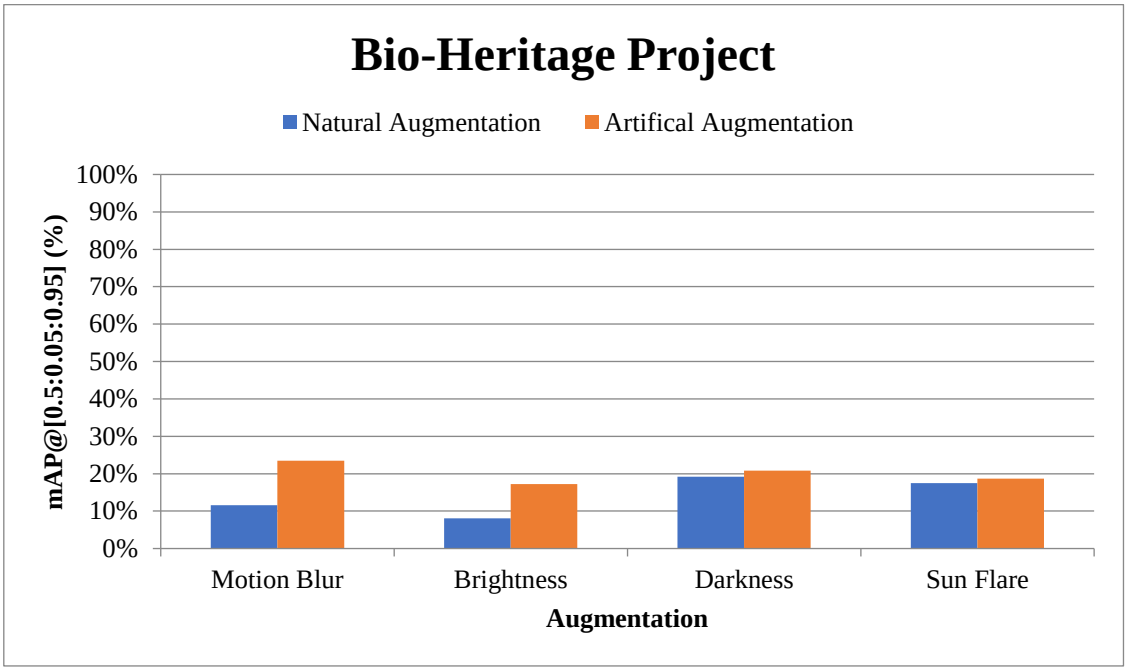


Figure 5.12 Graph showing the performance difference between naturally augmented images and artificially augmented images when evaluated on the Bio-Heritage model.

An ideal graph would show minimal difference between the natural and artificial augmentation performance as no difference would indicate that the model performed exactly the same on the artificial and natural augmentations. Simply put, no difference in performance indicates the model cannot distinguish between the natural and artificial augmentations. In Figure 5.12, the sun flare augmentation has the smallest performance difference, suggesting that out of the four augmentations, the artificial sun flare augmentation offers the closest substitution to its naturally occurring form.

Figure 5.13 Sample images from Bio-Heritage dataset comparing artificial and natural augmentations. Unaugmented images adapted from [102]

Interestingly, despite the sun flare augmentation providing the best substitution according to Figure 5.12, the sun flare augmentation shown in Figure 5.13 is the augmentation that looks the least like its natural counterpart, possibly suggesting that either there is another factor influencing the values shown in the graph, or that an accurate visual replication is not required to recreate the impact of a natural augmentation on an image.

A consistent challenge while creating this dataset, which is visible in Figure 5.13, was deciding between classifying natural augmentations as either brightness or sun flare. In the end, no definitive method was established to determine this, however in most cases this was determined by whether the augmentation was applied to the whole image or to regions of the image. Both the natural brightness and sun flare images provided in Figure 5.13 have this issue where they effectively have two augmentations. It is unknown to what extent this would have impacted the graph in Figure 5.12, but does indicate future research should explore the impact of multiple augmentations.

Another potential influencer in these results was the fact that while images were generally almost identical, most images are still taken from slightly different perspectives or at different times which does, to a minor degree, change the whole image. The impact of this is expected to be negligible, however could explain the performance difference between the two motion blur augmentations shown in Figure 5.13.

Overall, the difference in performance shown in Figure 5.12 leans towards the possibility of artificially generated augmentations becoming suitable substitutions for natural augmentations. It appears that accurate visual replication is not required for substitution, however, what is exactly required is not studied in this thesis. Furthermore, until now, the presence of multiple augmentations has been ignored. The images shown in Figure 5.13 however highlight the unavoidability of multiple augmentations in natural environments, suggesting a direction for future study.

5.4. Summary

This chapter began by proposing three questions which were explored through subsections 5.1, 5.2, and 5.3 respectively. Although the evaluations performed do not support a definitive answer to whether or not artificially generated augmentations are a viable substitute for naturally generated augmentations, the evaluations do partly support the answer to each of the three questions posed at the beginning of this chapter.

Impact. In Section 5.1, it was found that with the exception of a augmentations, artificial augmentations do impact the performance of a non-robust model. This was most clearly seen in the results for the Open Images V6 Trees datasets which had each augmentation decrease performance in a consistent manner.

Improve. In Section 5.2, it was shown that for seven out of the eight models trained, that training models on augmented datasets tends to improve a model's robustness towards that augmentation. The DeepWeeds dataset showed better improvement between the non-augmented and artificially augmented trained models than the Wellington Camera Traps, likely because of the DeepWeeds better baseline performance. Interestingly, for both models, the sun flare and motion blur augmentations had a bigger improvement than either the brightness or darkness augmentations, suggesting that improvement cannot be determined by the perceived difficulty of an augmentation.

Replicate. In Section 5.3, a case study was performed using a new, specially created dataset which enabled the comparison of naturally augmented images to artificially augmented images. The results indicated that the sun flare and darkness augmentations were fairly good approximations of their natural counterparts, whereas the motion blur and brightness

augmentations were not. It was noted in this section that despite appearing to be the best replication, the sun flare augmentation was also the augmentation that looked the least like its natural augmentation. Finally, the difficult of obtaining single-augmentation, naturally augmented images was discussed, suggesting a need for future work to investigate application of multiple augmentations to an image, making for more accurate substitutions.

6. Conclusion

This thesis has explored the creation and evaluation of artificially generated image augmentations in order to validate the usage of artificially generated augmentations as a substitute for naturally occurring augmentations for the purposes of improving deep learning based image recognition robustness.

Robustness is important for deep learning based computer vision models as models are typically bad at generalising objects they learn, meaning that any augmentation can interfere with a prediction. Robustness training through augmented datasets is one of many methods available for improving model robustness, but in many cases is the only applicable approach as the devices utilised in environments prone to augmentations tend to be low-powered edge devices that do not have the capability or resources to utilise other approaches.

To evaluate the substitution of the nine artificially generated augmentations discussed in Chapter 4 in place of their naturally occurring counterparts, Chapter 5 explored three questions investigating whether each augmentation impacts the performance of a non-robust model, improves the performance of a non-robust model and replicates the performance of a natural replication. These questions directed exploration of augmentation substitution by testing if qualities expected from naturally occurring augmentations were reproduced through each artificially generated augmentation.

Section 5.1 compared the performance impact over five increasingly stronger intensities of each augmentation to the baselines for each dataset as established in Chapter 3. It was observed in this section that artificially generated augmentations do, in most cases, reduce model performance in proportion to intensity.

Section 5.2 evaluated the performance improvement for the brightness, darkness, sun flare and motion blur augmentations at a mixed intensity on the DeepWeeds and Wellington Camera Traps datasets. This was assessed by comparing the F1-Score of two models: the baseline model and a model supplementally trained on images artificially augmented with the augmentation being evaluated. It was observed that for seven of the eight models trained, the supplemental training using artificially generated images resulted in an improvement over the performance of the baseline model. This improvement suggests that the artificially generated augmentations used in this thesis, for the most part, do improve generalisation within a model,

further suggesting that these augmentations are beneficial for robustness training. The most significant observation from this section was that performance improvement from artificial augmentation cannot be determined by the perceived strength of an augmentation.

Section 5.3 created artificially generated replicas of a handful of naturally occurring augmentations from a dataset derived from the Bio-Heritage project to evaluate whether a model trained on unaugmented images from this dataset would perform differently between naturally occurring and artificially generated augmentations. Of the four augmentations evaluated, only two augmentations had a similar performance impact between the naturally occurring and artificially generated versions. This suggested that some of the artificially generated augmentations used in this paper were not representative of their naturally occurring appearance in this dataset. This section also supports the observation from Section 5.2 in that performance impact between a naturally occurring and artificially generated augmentation cannot be determined by the perceived strength of the augmentation.

Overall, it was observed that while the artificially generated augmentations discussed in this paper appear to be substitutable for naturally occurring augmentations, attention must be paid to accurately replicating the augmentations, as it has been observed that creation of a visually similar artificially generated augmentation may be insufficient to accurately replicate a naturally occurring augmentation.

One difficulty for artificial replication that is not addressed in this thesis is the application of multiple artificial augmentations within an image. While applying multiple augmentations independently is trivial, naturally occurring augmentations are not applied individually and as a result are likely to exhibit emergent features as a result of this interaction. Images with multiple natural augmentations were avoided where possible within this thesis, however, must be considered for any future work to fully explore artificially generated augmentations as a substation for naturally generated augmentations.

Augmentation pre-set configuration is a challenge introduced by the implementation of the artificially generated augmentations within this thesis. Specifically, pre-sets in this thesis have been created such that they generate augmentations that appear to visually replicate their naturally occurring appearance. However, as it has been observed that creating visually similar artificially generated augmentations does not necessarily accurately replicate an augmentation, it might be worth exploring the usage of deep learning to create pre-sets. Using a collection of images containing a naturally occurring augmentation, a deep learning model could be created

to configure pre-set values that enable the creation of artificially generated augmentations that best match how a model perceives an augmentation. An alternative to this would be to create a deep learning model that generates artificial augmentations. This alternative would likely remove the need for the pre-set configurations; however, this approach requires a significant number of images containing naturally occurring augmentations, making it a very impractical alternative.

The findings from Chapter 5 neither definitively support nor reject the usage of artificially generated augmentations as substitutions for naturally occurring augmentations. However, they support their viability and indicate directions for future work that may produce more conclusive findings. In summary, this thesis has identified potential for using artificially generated augmentations as substitutions for naturally occurring augmentations, thereby supporting the usage of artificially generated augmentations for the creation of augmented robustness training datasets intended for improving model generalisation.

7. References

[1] H. Yang, L. Chen, Z. Ma, M. Chen, Y. Zhong, F. Deng, and M. Li, "Computer vision-based high-quality tea automatic plucking robot using Delta parallel manipulator," *Computers and Electronics in Agriculture,* vol. 181, p. 105946, 2021.

[2] T. Arnold. "Measurement in a flash." https://dhl-freight-connections.com/en/trends/measurement-in-a-flash/ (accessed 28/02/2023, 2023).

[3] J. Ravničan, A. Marinko, G. Noveski, S. Kalabakov, M. Jovanovič, S. Gazvoda, and M. Gams, "A Prestudy of Machine Learning in Industrial Quality Control Pipelines," *Informatica (03505596),* vol. 46, no. 2, 2022.

[4] J. Yang, S. Li, Z. Wang, H. Dong, J. Wang, and S. Tang, "Using deep learning to detect defects in manufacturing: a comprehensive survey and current challenges," *Materials,* vol. 13, no. 24, p. 5755, 2020.

[5] M.-H. Guo, T.-X. Xu, J.-J. Liu, Z.-N. Liu, P.-T. Jiang, T.-J. Mu, S.-H. Zhang, R. R. Martin, M.-M. Cheng, and S.-M. Hu, "Attention mechanisms in computer vision: A survey," *Computational Visual Media,* vol. 8, no. 3, pp. 331-368, 2022.

[6] C. Wang, C. Xu, C. Wang, and D. Tao, "Perceptual adversarial networks for image-to-image transformation," *IEEE Transactions on Image Processing,* vol. 27, no. 8, pp. 4066-4079, 2018.

[7] J. Han, M. Shoeiby, L. Petersson, and M. A. Armin, "Dual contrastive learning for unsupervised image-to-image translation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 746-755.

[8] G. R. Machado, E. Silva, and R. R. Goldschmidt, "Adversarial machine learning in image classification: A survey toward the defender's perspective," *ACM Computing Surveys (CSUR),* vol. 55, no. 1, pp. 1-38, 2021.

[9] R. A. Abumalloh, M. Nilashi, M. Y. Ismail, A. Alhargan, A. Alghamdi, A. O. Alzahrani, L. Saraireh, R. Osman, and S. Asadi, "Medical image processing and COVID-19: a literature review and bibliometric analysis," *Journal of infection and public health,* vol. 15, no. 1, pp. 75-93, 2022.

[10] M. Xu, S. Yoon, A. Fuentes, and D. S. Park, "A comprehensive survey of image augmentation techniques for deep learning," *Pattern Recognition,* p. 109347, 2023.

[11] R. Nirthika, S. Manivannan, A. Ramanan, and R. Wang, "Pooling in convolutional neural networks for medical image analysis: a survey and an empirical study," *Neural Computing and Applications,* vol. 34, no. 7, pp. 5321-5347, 2022.

[12] S. Suganyadevi, V. Seethalakshmi, and K. Balasamy, "A review on deep learning in medical image analysis," *International Journal of Multimedia Information Retrieval,* vol. 11, no. 1, pp. 19-38, 2022.

[13] Z. Zou, Z. Shi, Y. Guo, and J. Ye, "Object detection in 20 years: A survey," *arXiv preprint arXiv:1905.05055,* 2019.

[14] S. S. A. Zaidi, M. S. Ansari, A. Aslam, N. Kanwal, M. Asghar, and B. Lee, "A survey of modern deep learning based object detection models," *Digital Signal Processing,* p. 103514, 2022.

[15] OpenCV Team. "OpenCV - Open Computer Vision Library." https://opencv.org/ (accessed 12/02/2023, 2023).

[16] TensorFlow. "TensorFlow." https://www.tensorflow.org/ (accessed 12/02/2023, 2023).

[17] The Linux Foundation. "PyTorch." https://pytorch.org/ (accessed 12/02/2023, 2023).

[18] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research,* vol. 15, no. 1, pp. 1929-1958, 2014.

[19] W. Z. Khan, E. Ahmed, S. Hakak, I. Yaqoob, and A. Ahmed, "Edge computing: A survey," *Future Generation Computer Systems,* vol. 97, pp. 219-235, 2019.

[20] N. Gupta, "Artificial neural network," *Network and Complex Systems,* vol. 3, no. 1, pp. 24-28, 2013.

[21] OpenClipart. "Image: Artifical Neuron." https://freesvg.org/artificial-neuron (accessed 18/02/23, 2023).

[22] A. Chakarov, A. Nori, S. Rajamani, S. Sen, and D. Vijaykeerthy, "Debugging machine learning tasks," *arXiv preprint arXiv:1603.07292,* 2016.

[23] T. Zhang, C. Gao, L. Ma, M. Lyu, and M. Kim, "An empirical study of common challenges in developing deep learning applications," in *2019 IEEE 30th International Symposium on Software Reliability Engineering (ISSRE)*, 2019: IEEE, pp. 104-115.
[24] P. E. Rauber, S. G. Fadel, A. X. Falcao, and A. C. Telea, "Visualizing the hidden activity of artificial neural networks," *IEEE transactions on visualization and computer graphics,* vol. 23, no. 1, pp. 101-110, 2016.
[25] G. D. Cantareira, E. Etemad, and F. V. Paulovich, "Exploring neural network hidden layer activity using vector fields," *Information,* vol. 11, no. 9, p. 426, 2020.
[26] L. M. Zintgraf, T. S. Cohen, and M. Welling, "A new method to visualize deep neural networks," *arXiv preprint arXiv:1603.02518,* 2016.
[27] M. A. Nielsen, *Neural networks and deep learning*. Determination press San Francisco, CA, USA, 2015.
[28] L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M. A. Fadhel, M. Al-Amidie, and L. Farhan, "Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions," *Journal of big Data,* vol. 8, pp. 1-74, 2021.
[29] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature,* vol. 521, no. 7553, pp. 436-444, 2015.
[30] G. Marcus, "Deep learning: A critical appraisal," *arXiv preprint arXiv:1801.00631,* 2018.
[31] S. Liu and W. Deng, "Very deep convolutional neural network based image classification using small training sample size," in *2015 3rd IAPR Asian conference on pattern recognition (ACPR)*, 2015: IEEE, pp. 730-734.
[32] X. Ying, "An overview of overfitting and its solutions," in *Journal of physics: Conference series*, 2019, vol. 1168: IOP Publishing, p. 022022.
[33] T.-Y. Lin, G. Patterson, M. R. Ronchi, Y. Cui, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, L. Zitnick, and P. Dollár. "Coco Dataset,." https://cocodataset.org/#home (accessed 24/08/2022, 2022).
[34] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE,* vol. 86, no. 11, pp. 2278-2324, 1998.
[35] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, 2009: Ieee, pp. 248-255.
[36] ImageNet. "ImageNet Object Localization Challenge | Kaggle." Kaggle. https://www.kaggle.com/c/imagenet-object-localization-challenge/overview/description (accessed 10/03/2023, 2023).
[37] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International journal of computer vision,* vol. 88, pp. 303-308, 2009.
[38] M. V. G. Everingham, L; Williams, C. K. I; Winn, J; Zisserman, A. "The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results." http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html (accessed 10/03/2023, 2023).
[39] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," 2009.
[40] A. Krizhevsky. "CIFAR-10 and CIFAR-100 datasets." https://www.cs.toronto.edu/~kriz/cifar.html (accessed 10/03/2023, 2023).
[41] D. T. Krasin I., Alldrin N., Ferrari V., Abu-El-Haija S., Kuznetsova A., Rom H., Uijlings J., Popov S., Kamali S., Malloci M., Pont-Tuset J., Veit A., Belongie S., Gomes V., Gupta A., Sun C., Chechik G., Cai D., Feng Z., Narayanan D., Murphy K. "OpenImages: A public dataset for large-scale multi-label and multi-class image classification." https://storage.googleapis.com/openimages/web/index.html (accessed 10/03/2023, 2023).
[42] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM,* vol. 60, no. 6, pp. 84-90, 2017.
[43] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556,* 2014.
[44] F. N. Iandola, M. W. Moskewicz, K. Ashraf, and K. Keutzer, "Firecaffe: near-linear acceleration of deep neural network training on compute clusters," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2592-2600.
[45] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770-778.

[46] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818-2826.
[47] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Proceedings of the AAAI conference on artificial intelligence*, 2017, vol. 31, no. 1.
[48] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1492-1500.
[49] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510-4520.
[50] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8697-8710.
[51] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International conference on machine learning*, 2019: PMLR, pp. 6105-6114.
[52] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, and S. Gelly, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929,* 2020.
[53] H. Liu, Z. Dai, D. So, and Q. V. Le, "Pay attention to mlps," *Advances in Neural Information Processing Systems,* vol. 34, pp. 9204-9215, 2021.
[54] T.-Y. Lin, G. Patterson, M. R. Ronchi, Y. Cui, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, L. Zitnick, and P. Dollár. "COCO - Common Objects in Context." https://cocodataset.org/#detection-eval (accessed 24/08/2022, 2022).
[55] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," presented at the 2014 IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 2014.
[56] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440-1448.
[57] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779-788.
[58] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single Shot MultiBox Detector," Cham, 2016: Springer International Publishing, in Computer Vision – ECCV 2016, pp. 21-37.
[59] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961-2969.
[60] Z. Cai and N. Vasconcelos, "Cascade r-cnn: Delving into high quality object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 6154-6162.
[61] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934,* 2020.
[62] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: Scalable and efficient object detection," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 10781-10790.
[63] Y. Chen, Z. Zhang, Y. Cao, L. Wang, S. Lin, and H. Hu, "Reppoints v2: Verification meets regression for object detection," *Advances in Neural Information Processing Systems,* vol. 33, pp. 5621-5631, 2020.
[64] P. Sun, R. Zhang, Y. Jiang, T. Kong, C. Xu, W. Zhan, M. Tomizuka, L. Li, Z. Yuan, and C. Wang, "Sparse r-cnn: End-to-end object detection with learnable proposals," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 14454-14463.
[65] A. Ramanathan, L. Pullum, Z. Husein, S. Raj, N. Torosdagli, S. Pattanaik, and S. K. Jha, "Adversarial attacks on computer vision algorithms using natural perturbations," in *2017 Tenth International Conference on Contemporary Computing (IC3)*, 2017: IEEE, pp. 1-6.
[66] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199,* 2013.
[67] S. Dodge and L. Karam, "Understanding how image quality affects deep neural networks," in *2016 eighth international conference on quality of multimedia experience (QoMEX)*, 2016: IEEE, pp. 1-6.

[68] K. Sun, Z. Zhu, and Z. Lin, "Enhancing the robustness of deep neural networks by boundary conditional gan," *arXiv preprint arXiv:1902.11029,* 2019.
[69] D. Hendrycks and T. Dietterich, "Benchmarking neural network robustness to common corruptions and perturbations," *arXiv preprint arXiv:1903.12261,* 2019.
[70] S. Zheng, Y. Song, T. Leung, and I. Goodfellow, "Improving the robustness of deep neural networks via stability training," in *Proceedings of the ieee conference on computer vision and pattern recognition*, 2016, pp. 4480-4488.
[71] S. Diamond, V. Sitzmann, F. Julca-Aguilar, S. Boyd, G. Wetzstein, and F. Heide, "Dirty pixels: Towards end-to-end image processing and perception," *ACM Transactions on Graphics (TOG),* vol. 40, no. 3, pp. 1-15, 2021.
[72] A. Kar, S. K. Dhara, D. Sen, and P. K. Biswas, "Zero-shot single image restoration through controlled perturbation of koschmieder's model," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 16205-16215.
[73] J. Yim and K.-A. Sohn, "Enhancing the performance of convolutional neural networks on quality degraded datasets," in *2017 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, 2017: IEEE, pp. 1-8.
[74] T. S. Borkar and L. J. Karam, "DeepCorrect: Correcting DNN models against image distortions," *IEEE Transactions on Image Processing,* vol. 28, no. 12, pp. 6022-6034, 2019.
[75] E. Wong and J. Z. Kolter, "Learning perturbation sets for robust machine learning," *arXiv preprint arXiv:2007.08450,* 2020.
[76] N. Mu and J. Gilmer, "Mnist-c: A robustness benchmark for computer vision," *arXiv preprint arXiv:1906.02337,* 2019.
[77] D. Hendrycks. "Benchmarking Neural Network Robustness to Common Corruptions and Perturbations." https://github.com/hendrycks/robustness (accessed 25/01/2023, 2023).
[78] H. Nada, V. A. Sindagi, H. Zhang, and V. M. Patel, "Pushing the limits of unconstrained face detection: a challenge dataset and baseline results," in *2018 IEEE 9th International Conference on Biometrics Theory, Applications and Systems (BTAS)*, 2018: IEEE, pp. 1-10.
[79] H. S. Nada, Vishwanath; Zhang, He; Patel, Vishal M. "Unconstrained Face Detection Dataset (UFDD)." https://ufdd.info/ (accessed 2022-10-04, 2022).
[80] Adobe. "Photo Effects with Photoshop - How to do it - Adobe." https://www.adobe.com/products/photoshop/photo-effects.html (accessed 20/02/2023, 2023).
[81] H. Wang, Z. Yue, Q. Xie, Q. Zhao, Y. Zheng, and D. Meng, "From rain generation to rain removal," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 14791-14801.
[82] Z. Yue, J. Xie, Q. Zhao, and D. Meng, "Semi-supervised video deraining with dynamical rain generator," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 642-652.
[83] Y. Ye, Y. Chang, H. Zhou, and L. Yan, "Closing the loop: Joint rain generation and removal via disentangled image translation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 2053-2062.
[84] M. Uricar, G. Sistu, H. Rashed, A. Vobecky, V. R. Kumar, P. Krizek, F. Burger, and S. Yogamani, "Let's get dirty: Gan based data augmentation for camera lens soiling detection in autonomous driving," in *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, 2021, pp. 766-775.
[85] C. Michaelis, B. Mitzkus, R. Geirhos, E. Rusak, O. Bringmann, A. S. Ecker, M. Bethge, and W. Brendel, "Benchmarking robustness in object detection: Autonomous driving when winter is coming," *arXiv preprint arXiv:1907.07484,* 2019.
[86] M. D. Bloice, C. Stocker, and A. Holzinger, "Augmentor: an image augmentation library for machine learning," *arXiv preprint arXiv:1708.04680,* 2017.
[87] L. Jöckel, M. Kläs, and S. Martínez-Fernández, "Safe traffic sign recognition through data augmentation for autonomous vehicles software," in *2019 IEEE 19th International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, 2019: IEEE, pp. 540-541.
[88] Keras. "Keras: Deep Learning for Humans." https://keras.io/ (accessed 07/06/2023, 2023).
[89] A. Olsen, D. A. Konovalov, B. Philippa, P. Ridd, J. C. Wood, J. Johns, W. Banks, B. Girgenti, O. Kenny, and J. Whinney, "DeepWeeds: A multiclass weed species image dataset for deep learning," *Scientific reports,* vol. 9, no. 1, p. 2058, 2019.
[90] V. Anton, S. Hartley, A. Geldenhuis, and H. U. Wittmer, "Monitoring the mammalian fauna of urban areas using remote cameras and citizen science," *Journal of Urban Ecology,* vol. 4, no. 1, p. juy002, 2018.

[91] S. H. Victor Anton, Andre Geldenhuis, Heiko U Wittmer. *Wellington Camera Traps*. [Online]. Available: https://lila.science/datasets/wellingtoncameratraps
[92] B. M. Shashidhara, D. Mehta, Y. Kale, D. Morris, and M. Hazen, "Sequence Information Channel Concatenation for Improving Camera Trap Image Burst Classification," *arXiv preprint arXiv:2005.00116,* 2020.
[93] D. T. Krasin I., Alldrin N., Ferrari V., Abu-El-Haija S., Kuznetsova A., Rom H., Uijlings J., Popov S., Kamali S., Malloci M., Pont-Tuset J., Veit A., Belongie S., Gomes V., Gupta A., Sun C., Chechik G., Cai D., Feng Z., Narayanan D., Murphy K. "OpenImages: A public dataset for large-scale multi-label and multi-class image classification, 2017." https://storage.googleapis.com/openimages/web/index.html (accessed 2023).
[94] Voxel51 Inc. "Voxel51 // Open source computer vision tools for machine learning." https://voxel51.com/ (accessed 18/04/2023, 2023).
[95] Google LLC. *Open Images V6*. [Online]. Available: https://storage.googleapis.com/openimages/web/download_v6.html
[96] N. Bartlett. "COLOR TEMPERATURE CONVERSION OF HOMESTAR.IO." http://web.archive.org/web/20220618201011/https://www.zombieprototypes.com/?p=210 (accessed 24/04/2023.
[97] L. Bebber. "The Gooey Effect." https://css-tricks.com/gooey-effect/ (accessed 12/10/2022, 2022).
[98] J. Canny, "A computational approach to edge detection," *IEEE Transactions on pattern analysis and machine intelligence,* no. 6, pp. 679-698, 1986.
[99] OpenCV. "OpenCV: Feature Detection." https://docs.opencv.org/4.x/dd/d1a/group__imgproc__feature.html#ga04723e007ed888ddf11d9ba04e2232de (accessed 10/04/2023, 2023).
[100] A. Olsen, D. A. Konovalov, B. Philippa, P. Ridd, J. C. Wood, J. Johns, W. Banks, B. Girgenti, O. Kenny, and J. Whinney. *DeepWeeds*. [Online]. Available: https://github.com/AlexOlsen/DeepWeeds#download-the-dataset-images-and-our-trained-models
[101] D. L. Chin. "Optical Remote Sensing." https://web.archive.org/web/20120702174159/http://www.crisp.nus.edu.sg/~research/tutorial/optical.htm (accessed 23/10/2022, 2022).
[102] M. B. Dhami, Simon; Cridge, Andrew; Wilkinson, Shuan; Robertson, Blair; Green, Richar; Vetrova, Varvara; Kuang, Ye Chow; Ooi, Melanie; Duke, Mike, "Spectral Imaging for urban tree health," 2019 - 2024.
[103] Biological Heritage. https://bioheritage.nz/ (accessed 7/06/2023, 2023).

8. Appendices

A. Table of fixes for issues encountered with DeepWeeds training program.

Issue	Message	Mitigation	Reason
Requests library missing	ModuleNotFoundError: No module named 'requests'	Install package ‘requests’ through python package manager	Package missing from requirements.txt
Program downloads invalid ZIP files from Google Drive	zipfile.BadZipFile: File is not a zip file	Add: 'confirm': 'True' To line 61 inside params array in function call: session.get()	Google Drive no longer appears to require a confirmation code which program attempts to supply. Setting confirm parameter to true is sufficient
Invalid Y column datatype	TypeError: If class_mode="categorical", y_col="Label" column values must be type string, list or tuple.	add argument ‘`dtype=str`’ for lines 166,167 and 168 to `read_csv()` function call.	Pandas library seems to be interpreting CSV column types incorrectly
Images not loaded into pipeline	Found 0 validated image filenames belonging to 9 classes.	Under variable on line 44 CLASSES Add variable CLASSES_STR = list(map(str,CLASSES)) Replace CLASSES With CLASSES_STR In all functions flow_from_dataframe()	Fallout result from the previous mitigation, CSV now loads classes as strings which do not match integer array.
Issue creating InceptionV3 model through Keras	AttributeError: 'str' object has no attribute 'decode'	Install version 2.10.0 of h5py through python package manager	Issue appears to occur with newer versions of h5py. Resolved by downgrading

B. Per-Fold values for Baseline evaluations

TRAINING + VALIDATION SET	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
DeepWeeds (F1-Score)	0.934	0.932	0.932	0.936	0.938	**0.934**
Wellington Camera Traps (F1-Score)	0.615	0.611	0.581	0.595	0.630	**0.606**
Open Images V6 Trees (mAP)	0.210	0.229	0.301	0.118	0.216	**0.215**

TEST SET	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
DeepWeeds (F1-Score)	0.945	0.954	0.952	0.960	0.955	**0.953**
Wellington Camera Traps (F1-Score)	0.556	0.577	0.666	0.590	0.634	**0.605**
Open Images V6 Trees (mAP)	0.307	0.331	0.179	0.125	0.257	**0.240**

C. FiftyOne function used to download Tree Subset of OpenImages V6

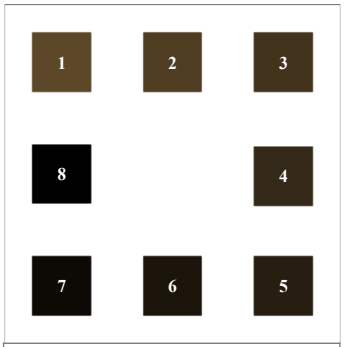
```
dataset = fiftyone.zoo.load_zoo_dataset(
              "open-images-v6",
              split=["train", "test", "validation"],
              label_types=["detections",],
              classes=["Tree”],
              max_samples=15000,
          )
```

D. ColourPalette type – Dirt Augmentation

The colour palette is a custom type created to provide the dirt augmentation with a collection of eight colours without requiring the user to specify all eight colours individually. There is no restriction on what colours can be defined however the type must contain eight colours as the type does not support any other number of colours. In the dirt augmentation, each polygon drawn has a colour randomly selected from the provided palette.

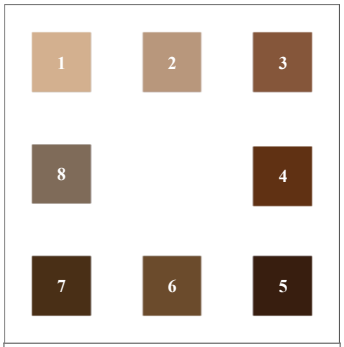
The class comes with three "known palettes": "Palette1", "Palette2", and "Palette3". These palettes are configured as shown in the image below.

Palette1



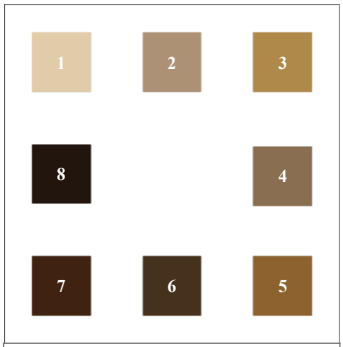
Property	Value (Hex)
Colour1	#5C4728
Colour2	#4E3D21
Colour3	#42331C
Colour4	#342919
Colour5	#271D11
Colour6	#1B140A
Colour7	#0D0A05
Colour8	#000000

Palette2



Property	Value (Hex)
Colour1	#D3B08F
Colour2	#B8977C
Colour3	#855A3A
Colour4	#603113
Colour5	#381E0F
Colour6	#6B4B2C
Colour7	#492F16
Colour8	#7F6B59

Palette3



Property	Value (Hex)
Colour1	#E2CBA9
Colour2	#AC9174
Colour3	#AF894A
Colour4	#896E51
Colour5	#8D622F
Colour6	#46311C
Colour7	#3F2210
Colour8	#22150D

Custom colour palettes can be created in an augmentation either by creating a new instance of the ColourPalette class with desired colour values or through conversion of a comma separated string of a base 10 ARGB colour.

To enable randomisation when using pre-set configurations, a colour palette converter type was created. The convert can be used in a pre-set by setting the value of the type XML attribute.

```
Type="DIRTFX::COLOURPALETTE"
```

As maximum and minimum do not make sense in the context of a collection of colours, only the minimum attribute is required. For randomisation, the colour palette type converter expects

a semicolon separated string containing either “known palette” names or string-formatted custom palettes.

This type has been designed specifically for the dirt augmentation and therefore is packaged with the ‘AAG_Dirt plugin.

E. Dirt Augmentation Particle Simulation Pseudo-Code

```
Class Point2:
        Float    X
        Float    Y
```

```
Class Point3:
        Float    X
        Float    Y
        Float    Z
```

```
Class Particle:
        Point3   Direction
        Point3   Position
        Float    Weight
        Float    Speed
        Boolean  HasCollided
```

```
        Function New(int worldmaxX, int worldmaxY, int worldmaxZ) returns Particle:
                Set Direction = RandFloat(0, worldmaxx…)
                Set Weight = RandFloat(1.0, 5.0)
                Set Speed = RandFloat(10.0, 40.0)
                Set HasCollided = False
                Reutrn Particle;
```

```
Function Simulate (int maxRounds, int particlecount, Point3 origin, int worldmaxX, int worldmaxY, int worldmaxZ) returns Point2[]:

        Set particles = Particle.New(worldmaxX, worldmaxY, worldmaxZ) x [particlecount]

        Foreach round = 1 to maxRounds:

                Foreach particle in particles:

                        If particle.Direction.Z < 0 and particle.HasCollided = False:

                                Set particle.HasCollided = True

                        If particle.HasCollided = True:

                                Foreach axis (x,y,z) in particle.Position:

                                        If particle.Position[axis] >= worldmax(axis) or <= 0:

                                                Set particle.Direction[axis] = RandFloat(0.0, 1.0) * -1 * particle.Direction[axis]

                                                Set particle.Speed = particle.Speed / (3 * particle.Weight)

                        Else:

                                Foreach axis (x,y,z) in particle.Position:

                                        If particle.Position[axis] = origin[axis]:

                                                particle.Direction[axis] = 0

                                        Else If particle.Position[axis] < origin[axis]:

                                                particle.Direction[axis] = RandFloat(0.0, 1.0)

                                        Else:

                                                particle.Direction[axis] = RandFloat(0.0, 1.0) * -1

                        Set particle.Position = particle.Position + (particle.Direction * particle.Speed * particle.Weight)

        Set newParticles = Point2[]

        Foreach particle in particles:

                Add (particle.Position.X, particle.Position.Y) to newParticles

        Return newParticles
```

F. Colour temperature to RGB

```
public static class SunFlareKelvinConverter
{
	readonly static double[] a = new double[] { 351.97690566805693, -155.25485562709179,
	325.4494125711974, -254.76935184120902 };
	readonly static double[] b = new double[] { 0.1142064537841 65, -0.44596950469579133,
	0.07943456536662342, 0.8274096064007395 };
	readonly static double[] c = new double[] { -40.25366309332127, 104.49216199393888, -
	28.0852963507957, 115.67994401066147 };

	public static Color FromKelvin(int k)
	{
		int nr = 0;
		int ng = 0;
		int nb = 0;

		if (k >= 6600)
		{
			double x_r = (k / 100) - 55;
			double x_g = (k / 100) - 50;
			nr = (int)(a[0] + b[0] * x_r + c[0] * Math.Log(x_r));
			ng = (int)(a[2] + b[2] * x_g + c[2] * Math.Log(x_g));
			nb = 255;
		}
		else
		{
			double x_g = (k / 100) - 2;
			double x_b = (k / 100) - 10;
			nr = 255;
			ng = k > 1000 ? (int)(a[1] + b[1] * x_g + c[1] * Math.Log(x_g)) : 0;
			nb = k > 2000 ? (int)(a[3] + b[3] * x_b + c[3] * Math.Log(x_b)) : 0;
		}
		return Color.FromArgb(255, Clamp(nr,0,255), Clamp(ng, 0, 255), Clamp(nb, 0,
		255));

	}

	private static int Clamp(int x, int min, int max)
	{
		return Math.Max(min, Math.Min(max, x));
	}
}
```

G. Scaling constants used for water-based augmentations

Focused Rain, Unfocused Rain:

Scaling variable	Value
a	21.569
b	0.0006
c	10.178
d	0.0016

Frost

Scaling variable	Value
a	8.9917
b	0.0022
c	12.697
d	0.0037

Water Residue

Scaling variable	Value
a	6.9917
b	0.0022
c	6.6970
d	0.0037

H. Pre-Set Configurations

H-1. Focused Rain Pre-Sets

Wellington Camera Traps Pre-Set

INTENSITY		1		2		3		4		5	
PROP	**TYPE**	**min**	**max**	**min**	**max**	**min**	**max**	**min**	**max**	**min**	**max**
Drops	**I**	25	50	50	100	100	200	200	400	400	800
PROP	**TYPE**	**value**		**value**		**value**		**value**		**value**	
Background Blur	**I**	15		15		15		15		15	
Foreground Blur	**I**	1		1		1		1		1	
Refraction	**I**	20		20		20		20		20	
Tolerance	**I**	10		10		10		10		10	

Open Images V6 Trees Pre-Set

INTENSITY		1		2		3		4		5	
PROP	**TYPE**	**min**	**max**	**min**	**max**	**min**	**max**	**min**	**max**	**min**	**max**
Drops	**I**	25	50	50	100	100	200	200	400	400	800
PROP	**TYPE**	**value**		**value**		**value**		**value**		**value**	
Background Blur	**I**	15		20		20		25		25	
Foreground Blur	**I**	1		1		1		1		1	
Refraction	**I**	20		20		20		20		20	
Tolerance	**I**	10		10		10		10		10	

H-2. Unfocused Rain Pre-Sets

Standard Pre-Set

INTENSITY		1		2		3		4		5	
PROP	**TYPE**	**min**	**max**	**min**	**max**	**min**	**max**	**min**	**max**	**min**	**max**
Drops	**I**	25	50	50	100	100	200	200	400	400	800
Intensity	**I**	32	64	32	64	32	64	32	64	32	64
PROP	**TYPE**	**value**		**value**		**value**		**value**		**value**	
Foreground Blur	**I**	1		1		1		1		1	
Refraction	**I**	20		20		20		20		20	
Tolerance	**I**	10		10		10		10		10	

I. Per-Fold values for Impact tables

I-1.DeepWeeds

INTENSITY 1	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Dark	0.94	0.95	0.95	0.96	0.96
Dirt	0.95	0.95	0.95	0.96	0.95
Focused Rain	0.49	0.41	0.28	0.26	0.39
Frost	0.61	0.62	0.61	0.61	0.62
Light	0.95	0.95	0.95	0.96	0.95
Motion	0.92	0.92	0.91	0.92	0.92
Sun Flare	0.89	0.91	0.90	0.90	0.90
Unfocused Rain	0.80	0.77	0.77	0.78	0.80
Water Residue	0.94	0.94	0.94	0.94	0.94

INTENSITY 2	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Dark	0.93	0.94	0.93	0.94	0.95
Dirt	0.91	0.92	0.93	0.93	0.91
Focused Rain	0.43	0.35	0.22	0.22	0.35

Frost	0.50	0.51	0.48	0.48	0.54
Light	0.94	0.94	0.94	0.95	0.94
Motion	0.86	0.83	0.84	0.84	0.85
Sun Flare	0.68	0.75	0.72	0.74	0.71
Unfocused Rain	0.61	0.62	0.57	0.59	0.61
Water Residue	0.92	0.92	0.92	0.92	0.91

INTENSITY 3	**Fold 1**	**Fold 2**	**Fold 3**	**Fold 4**	**Fold 5**
Dark	0.86	0.88	0.87	0.87	0.88
Dirt	0.83	0.86	0.89	0.88	0.83
Focused Rain	0.38	0.31	0.21	0.20	0.30
Frost	0.27	0.33	0.35	0.28	0.30
Light	0.91	0.91	0.91	0.92	0.92
Motion	0.75	0.70	0.69	0.71	0.75
Sun Flare	0.21	0.19	0.19	0.21	0.23
Unfocused Rain	0.47	0.47	0.45	0.46	0.49
Water Residue	0.90	0.90	0.89	0.89	0.88

INTENSITY 4	**Fold 1**	**Fold 2**	**Fold 3**	**Fold 4**	**Fold 5**
Dark	0.70	0.71	0.69	0.71	0.71
Dirt	0.58	0.66	0.74	0.72	0.67
Focused Rain	0.35	0.26	0.20	0.19	0.30
Frost	0.16	0.20	0.20	0.16	0.19
Light	0.82	0.83	0.83	0.84	0.83
Motion	0.64	0.58	0.53	0.58	0.62
Sun Flare	0.16	0.14	0.15	0.16	0.18
Unfocused Rain	0.38	0.37	0.36	0.37	0.41
Water Residue	0.89	0.88	0.87	0.87	0.88

INTENSITY 5	**Fold 1**	**Fold 2**	**Fold 3**	**Fold 4**	**Fold 5**
Dark	0.46	0.52	0.51	0.53	0.49

Dirt	0.13	0.18	0.20	0.18	0.15
Focused Rain	0.48	0.40	0.39	0.39	0.46
Frost	0.17	0.16	0.17	0.13	0.18
Light	0.56	0.54	0.60	0.62	0.60
Motion	0.56	0.50	0.44	0.50	0.56
Sun Flare	0.13	0.12	0.13	0.14	0.16
Unfocused Rain	0.33	0.30	0.30	0.30	0.36
Water Residue	0.89	0.89	0.87	0.88	0.89

I-2.Wellington Camera Traps

INTENSITY 1	**Fold 1**	**Fold 2**	**Fold 3**	**Fold 4**	**Fold 5**
Dark	0.57	0.57	0.66	0.60	0.63
Dirt	0.56	0.58	0.66	0.59	0.64
Focused Rain	0.45	0.45	0.44	0.44	0.46
Frost	0.49	0.53	0.47	0.48	0.52
Light	0.56	0.58	0.66	0.60	0.64
Motion	0.57	0.53	0.58	0.53	0.57
Sun Flare	0.50	0.49	0.61	0.53	0.56
Unfocused Rain	0.54	0.57	0.64	0.58	0.63
Water Residue	0.52	0.56	0.60	0.57	0.60

INTENSITY 2	**Fold 1**	**Fold 2**	**Fold 3**	**Fold 4**	**Fold 5**
Dark	0.55	0.52	0.61	0.59	0.60
Dirt	0.55	0.58	0.63	0.57	0.61
Focused Rain	0.45	0.45	0.44	0.44	0.46
Frost	0.51	0.53	0.49	0.49	0.54
Light	0.55	0.57	0.64	0.57	0.61
Motion	0.51	0.49	0.49	0.46	0.54
Sun Flare	0.48	0.46	0.56	0.52	0.51
Unfocused Rain	0.52	0.56	0.62	0.58	0.62

Water Residue	0.52	0.55	0.58	0.54	0.59

INTENSITY 3	**Fold 1**	**Fold 2**	**Fold 3**	**Fold 4**	**Fold 5**
Dark	0.48	0.48	0.56	0.56	0.55
Dirt	0.53	0.54	0.52	0.49	0.60
Focused Rain	0.47	0.46	0.44	0.44	0.47
Frost	0.50	0.53	0.54	0.51	0.55
Light	0.52	0.52	0.61	0.55	0.57
Motion	0.48	0.48	0.45	0.45	0.48
Sun Flare	0.50	0.48	0.46	0.48	0.48
Unfocused Rain	0.51	0.56	0.57	0.55	0.60
Water Residue	0.52	0.53	0.53	0.55	0.58

INTENSITY 4	**Fold 1**	**Fold 2**	**Fold 3**	**Fold 4**	**Fold 5**
Dark	0.46	0.48	0.51	0.52	0.54
Dirt	0.55	0.50	0.50	0.46	0.51
Focused Rain	0.49	0.46	0.44	0.45	0.48
Frost	0.46	0.52	0.49	0.47	0.49
Light	0.48	0.48	0.58	0.53	0.56
Motion	0.46	0.47	0.45	0.45	0.48
Sun Flare	0.52	0.47	0.38	0.48	0.49
Unfocused Rain	0.50	0.54	0.52	0.52	0.55
Water Residue	0.50	0.51	0.49	0.53	0.58

INTENSITY 5	**Fold 1**	**Fold 2**	**Fold 3**	**Fold 4**	**Fold 5**
Dark	0.45	0.49	0.52	0.47	0.50
Dirt	0.49	0.47	0.48	0.46	0.47
Focused Rain	0.50	0.50	0.47	0.48	0.49
Frost	0.44	0.52	0.46	0.47	0.48
Light	0.46	0.47	0.52	0.53	0.50
Motion	0.44	0.47	0.44	0.45	0.47

Sun Flare	0.49	0.46	0.38	0.47	0.51
Unfocused Rain	0.50	0.50	0.47	0.46	0.49
Water Residue	0.49	0.51	0.47	0.51	0.56

I-3.Open Images V6. Trees

INTENSITY 1	**Fold 1**	**Fold 2**	**Fold 3**	**Fold 4**	**Fold 5**
Dark	0.30	0.32	0.18	0.12	0.25
Dirt	0.30	0.33	0.18	0.13	0.26
Focused Rain	0.14	0.16	0.11	0.10	0.13
Frost	0.09	0.11	0.06	0.05	0.06
Light	0.30	0.33	0.19	0.12	0.26
Motion	0.23	0.25	0.15	0.11	0.19
Sun Flare	0.29	0.32	0.18	0.12	0.25
Unfocused Rain	0.29	0.32	0.17	0.13	0.25
Water Residue	0.28	0.31	0.17	0.12	0.24

INTENSITY 2	**Fold 1**	**Fold 2**	**Fold 3**	**Fold 4**	**Fold 5**
Dark	0.27	0.30	0.17	0.12	0.24
Dirt	0.29	0.32	0.18	0.13	0.25
Focused Rain	0.09	0.13	0.08	0.08	0.09
Frost	0.05	0.07	0.05	0.03	0.04
Light	0.30	0.32	0.18	0.12	0.25
Motion	0.12	0.15	0.09	0.09	0.11
Sun Flare	0.28	0.30	0.17	0.12	0.24
Unfocused Rain	0.28	0.30	0.17	0.12	0.24
Water Residue	0.27	0.29	0.16	0.12	0.23

INTENSITY 3	**Fold 1**	**Fold 2**	**Fold 3**	**Fold 4**	**Fold 5**
Dark	0.22	0.24	0.14	0.11	0.20

Dirt	0.25	0.28	0.16	0.10	0.21
Focused Rain	0.07	0.10	0.07	0.05	0.08
Frost	0.02	0.03	0.02	0.01	0.01
Light	0.28	0.31	0.17	0.12	0.24
Motion	0.07	0.10	0.06	0.07	0.08
Sun Flare	0.20	0.24	0.14	0.11	0.20
Unfocused Rain	0.26	0.27	0.16	0.12	0.22
Water Residue	0.24	0.26	0.16	0.12	0.23

INTENSITY 4	**Fold 1**	**Fold 2**	**Fold 3**	**Fold 4**	**Fold 5**
Dark	0.16	0.18	0.12	0.09	0.15
Dirt	0.16	0.20	0.11	0.08	0.14
Focused Rain	0.03	0.04	0.03	0.02	0.04
Frost	0.01	0.02	0.01	0.00	0.01
Light	0.26	0.29	0.17	0.11	0.23
Motion	0.05	0.07	0.04	0.05	0.05
Sun Flare	0.17	0.22	0.12	0.10	0.17
Unfocused Rain	0.22	0.24	0.15	0.11	0.19
Water Residue	0.22	0.24	0.14	0.11	0.19

INTENSITY 5	**Fold 1**	**Fold 2**	**Fold 3**	**Fold 4**	**Fold 5**
Dark	0.12	0.13	0.09	0.08	0.11
Dirt	0.09	0.12	0.08	0.06	0.09
Focused Rain	0.03	0.03	0.03	0.02	0.04
Frost	0.01	0.01	0.01	0.00	0.01
Light	0.23	0.26	0.15	0.11	0.21
Motion	0.04	0.05	0.03	0.04	0.04
Sun Flare	0.15	0.19	0.11	0.09	0.15
Unfocused Rain	0.16	0.18	0.12	0.10	0.14
Water Residue	0.20	0.21	0.14	0.11	0.17