Title: Improving Domain Generalisation in the Task of Animal Detection in Camera Trap Data

G037 (s2255744, s2125423, s2213510)

Abstract

The use of automated object detection with motion triggered camera traps can potentially be a great tool for ecologists to help identify animals in the wild. While powerful deep neural networks and large labelled datasets are available, identifying animals at new arbitrary camera locations remains a challenge due to the bias of training data and poor generalisation performance on unseen image backgrounds. In this paper, we try to improve such generalisation problem by first training a benchmark detection model on the Caltech Camera Traps-20 (CCT-20) dataset as demonstrated in (Beery et al., 2018). Various techniques such as image augmentations and regularization are then explored and evaluated using images taken at locations not present in the training set, and consequently have different backgrounds. Our experiments show that gathering more training data improves the validation accuracy performance most effectively (from 40.2% to 47.1% mAp) even if the backgrounds in this extra data are from the same distribution. As well as this, the results can be further improved by random cropping data augmentation and L2 regularisation to mAp 49.9% and 50.8% respectively.

1. Introduction

Object detection with camera trap images is a challenge for many state-of-the-art object detection methods, due to the randomness of the images taken. Animals can be all sizes, distances from the camera, blurred, hidden, and many images are empty. This paper will be looking at ways to improve detection models on camera trap data, and improving generalisation to new domains with a relatively small dataset (~57,824 images in the Benchmark dataset vs 243,187 images in the original dataset).

The challenge of identifying animal species in camera trap images can be a long, tedious task for many researchers. Ecologists can use these images to identify animal populations and activity in certain regions, whilst being cheap and remaining unintrusive on these animals (Burton et al., 2015). Generally, large datasets are created using camera trap images, but they tend to be largely imbalanced with images that do not contain animals. Hence, it can be difficult for researchers to filter through thousands of images just to obtain photos that do not contain certain species that are of interest. Machine Learning methods have been increasingly used to identify animals and species in these photos, to drastically reduce the time spent filtering through these images. However, even using these methods can be difficult due to the range of different locations these traps are used in and the randomness of images for each species. Some datasets may contain a large number of instances of one species, but very little instances of other species. Furthermore, the challenge with these datasets is generalising to new locations (Beery et al., 2018). Hence, the main focus of this project will be looking at how we can improve the generalisation of models to increase the accuracy when testing on new locations. This is important so that these models can be robust and perform well on new datasets and locations that they have not been trained on. This could drastically save time for ecologists and those tasked with animal population monitoring or control.

We have chosen the task of detection over classification as it is said to perform and generalise better to new locations (Norouzzadeh et al., 2021), which is something we hope to further improve upon in our project. (Schneider et al., 2019) use similar methods to that which we look at in this paper, using Faster R-CNN and YOLO v2.0 on the Reconyx Camera Trap and the self-labelled Snapshot Serengeti datasets. (Vecvanags et al., 2022) create a new dataset and use Faster R-CNN and RetinaNet to detect animals, looking at data augmentation and optimisation. Another study by (Touh & Sharma, 2020) looked at using the Faster R-CNN with the Gold Standard Snapshot Serengeti Dataset. Using data augmentations, they went from around 4000 images to around 28000 images, vastly improving the mAp score. We look at other works relating to camera trap data in the related works section.

The primary goal of this project is to improve on the benchmark accuracy of the Faster R-CNN model by looking at different data pre-processing methods to hopefully improve on the accuracy and robustness of these models. A large part of improving performance is improving domain generalisation and this will be looked at in this paper. This will involve using data augmentation techniques and regularisation. We expect that this will improve the performance of the model, as it should help the model to adapt to randomness in the data and improve robustness of the model. We will be comparing Precision, Average Recall and loss across experiments to understand how these metrics change. The novelty in this project comes from the fact we are using the

Benchmark Caltech Camera Traps Dataset, which has not been used a lot in previous works. The additional challenge here is the small dataset size, meaning the smaller classes do not have a lot of examples to train on. The aim and the objectives for the project are presented below.

Aim:
Improve upon the performance of a baseline object detection model on a camera trap dataset with domain generalization techniques.

Objectives:

1. Implement Faster-RCNN and measure the validation performance on a baseline model
2. Measure the model performance with various augmentations on the dataset
3. Measure the model performance with various regularization methods (dropout, L1, L2)
4. Measure the model performance with combinations of these techniques
5. Present the performance of the baseline model and best model on the test set

2. Data set and task

The dataset we are using is the Benchmark Caltech Camera Traps-20 (CCT-20) Data subset (Beery et al., 2018). The original paper worked with the same set of images but in a higher resolution that is roughly double in both image height and width. Our dataset reduced the total storage size from 22GB to 6GB. This inevitably causes adverse effect to the model accuracy, but would speed up our experiments, thus allowing more generalisation methods to be explored. The dataset contains 57,864 images, split into training (13,553), validation (cis: 3,484; trans: 1,725) and test sets (cis: 15,827; trans: 23,275). This leaves us with a 41%, 11%, 48% split. The test set and validation set are further split into cis, locations seen in the training data, and trans, new locations not seen at training time. We will be using two baselines. One using the training data and evaluating on the trans. The other will combine the train and cis data into one large training set, and evaluating on trans. This is to see how increasing the training dataset size increases the accuracy. The idea is that we wish for our model to generalise well to new locations and so the performance on the trans dataset is especially important. There is a large inbalance of classes. For example, the train set includes 2751 Opossums, but only 9 Badgers. Across the training data and cis data there is 10 locations. The trans data uses 10 new locations, 1 location for the trans validation set and 9 locations in the trans test set. The class distribution across locations can be seen in Figure 2. The images are in JPG format and the JSON annotation files contain information such as the IDs of 16 categories, the locations, the width and height of the images, the IDs for mapping between annotations and images, the filename

(a)

(b)

Figure 1. (a) An image of two wild dogs (b) An infrared image of a rabbit at night

and the bounding box location (unless the image is empty). Some images may also contain a sequence number, which implies a sequence of images were taken at a single motion trigger. Some example images from the benchmark dataset are shown in Figure 1. Its not unusual to see images containing more than one animal, as in Figure 1(a). In this case, our model should be able to identify two separate bounding boxes. In some images in the testing set, animals may not be present, as a result of false trigger by wind or animals moving out of field of view during the image sequence.

In terms of data preprocessing, images must first be linked to their annotations, so that the bounding boxes can match with the locations of animals. In order to work with the Tensorflow API, due to the random category ID's of the animals, we had to map animals to ID's so that category ID's went from 1 to 16. TfRecords were then produced, to store the dataset efficiently and save space. After we have sorted the image-annotation pairs we further pre-process the data. (Beery et al., 2018) use cropping, horizontal flipping and colour distortion of images to augment their data for classification, but only use horizontal flipping for detection. Employing these other steps for detection may help to increase accuracy.

Precision and Recall are popular metrics in object detection tasks, as accuracy is sensitive to the imbalance in classes. This helps to look at how many false positives and false neg-

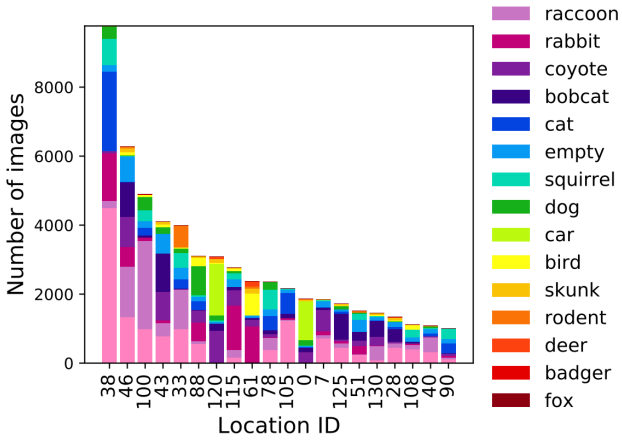

Figure 2. The class distribution per location ((taken from (Beery et al., 2018)

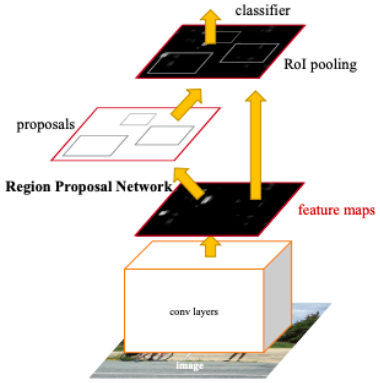

Figure 3. An overview of the Faster R-CNN network (taken from (Ren et al., 2015))

atives are in the data. We will also use the COCO metrics to analyse our results. Average precision is simply the area under the precision-recall curve. IoU, the Intersection over Union, is a measure of the overlap between two boundaries, scored between 0 and 1. This is used between the models predictions and the ground truth bounding boxes. In basic terms, this can be defined as

$$IoU = \frac{\text{Area of intersection}}{\text{Area of Union}}$$

mAp, the mean average precision, is the average precision over all classes and this is the most popular metric. The lower the IOU threshold, the more lenience the model displays in classifying the bounding box as correct.

$$mAP = \frac{1}{n}\sum_{k=1}^{k=n} AP_k$$

where AP_k is the average precision of class k and n is the number of classes.

This means if the model is bad at dealing with a particular class, this will decrease the mAp. We will be using mAp as the metric to measure how the performance in models changes.

3. Methodology

The main components of our work focus around the Faster R-CNN model with the ResNet-101 backbone (a CNN with 101 layers) and looking at how we can improve generalisation of this model. This involves looking at different data augmentation strategies and regularisation techniques such as Dropout and L2 regularisation.

3.1. Faster R-CNN

Faster R-CNN (Figure 3) is a state-of-the-art object detection model that has achieved impressive results on object detection datasets such as MS COCO and PASCAL VOC 2007, 2012 (Ren et al., 2015). It evolved from previous R-CNN methods such as R-CNN and Fast R-CNN, increasing the speed at which detections can be made by utilising a Region Proposal Network (RPN). It uses a deep convolutional network (the RPN network) to generate bounding box proposals and then the Fast R-CNN network to make classifications. Both networks share the same convolutional layers which is responsible for the increase in speed.

3.1.1. Region Proposal

A number of backbone models can be used as convolutional networks for this model. The model we are using uses ResNet. The input is fed into these networks and the RPN detects objects in these regions using an objectness score to output the likelihood of an object being in this box. These bounding boxes are called anchors and use different scales and aspect ratios to propose areas of the image where objects may be, which helps to account for the different sizes of objects in the images. Using the last convolutional feature map, a small network is used to slide over the this map in windows of size nxn. Each window is then fed into a lower-dimensional feature, which is fed into two fully connected layers called the box-regression layer and the box-classification layer. Within each sliding-window, a maximum of k region proposals are predicted. The box-regression layer outputs bounding boxes and the box-classification layer outputs object probabilities. The proposals are made with respect to the anchors, providing different scales and aspect ratios. The IoU is calculated across anchors to give each anchor a positive or negative score. This can be scaled by changing the IoU threshold. The use of attention in the RP network helps the Fast R-CNN to know where to focus to detect animals in this case.

By evaluating the general width to height ratios of animals in the dataset, we have used scales of 0.25, 0.5, 1.0 and 2.0 and anchor aspect ratios of 0.5, 1.0 and 2.0.

3.1.2. Training

Faster R-CNN uses the idea of alternate training in order to train the weights of the model (Ren et al., 2015). The

	Loss Function
CL_C	Classification Loss of Classifier
LL_C	Localization Loss of Classifier
LL_{RPN}	Localization Loss Of RPN
OL_{RPN}	Objectness Loss of RPN

Table 1. Loss Functions for Baseline Model

RPN is first trained in order to generate the region proposals. Using a pre-trained model on ImageNet, the shared convolutional layers are initialised and the other weights of the RPN are initilaised randomly. Once generated, the RPN and shared convolutional layers are tuned. These boxes are then used to train the Fast R-CNN module, using the same weights for the shared convolutional layers and the other weights initialised randomly. During training, the weights of both the Fast R-CNN and the shared convolutional layers are tuned. This cycle repeats again starting with the RPN. The loss functions are summarised in Table 1. The localization losses are the losses of bounding box regressor and objectness loss is the loss classifying whether a bounding box contains background or object. The classification loss is the loss classifying the correct animal and is the least interesting in this detection task.

3.2. Data Augmentation

A large part of data generalisation is down to using data augmentation strategies, allowing the model to learn to adapt to unseen data (Song et al.). This is especially the case in camera trap data, in which animals could be at random points in the image. These could be in relation to blur, proximity to camera, animals cropped at the edge of the photo, and time of day (brightness). It makes sense to analyse data augmentation strategies that are relevant to our problem, as using the wrong augmentation strategies can have negative impacts on downstream tasks (Song et al.). Therefore, we look to use horizontal flipping, cropping, brightness adjustment, patch gaussian, and a combination of these methods. We will further look at adjusting probability rates to look at what the best proportion of images in the dataset should be augmented to, to obtain best results.

Random horizontal flipping is a common transformation in a lot of problems, which simply mirrors the image in the vertical axis. It has been an effective method across a number of datasets such as CIFAR-10 and ImageNet (Shorten & Khoshgoftaar, 2019). This will prove useful in our dataset, as it is common that animals will be facing in different directions and appear on different sides of the photo.

The dataset also contains a large number of dark images which add further complications for the model. The hope is that by adding brightness, this can help to improve performance, particularly for darker images. The dataset is biased mostly towards day time images so hopefully the dataset will be less biased by constraining all images to the same brightness (Shorten & Khoshgoftaar, 2019). In this case, we use a value of 0.2, meaning image outputs will be changed by up to 0.2 brightness.

Random cropping is used to crop the image down to a smaller size. The hope is that this will cause the animal to appear at different parts of the image, and potentially parts of it cropped out of the photo. It may also help to focus more on the animal, and less on the background which will help performance when evaluating on the trans dataset. We do this by ensuring the cropped image covers at least one bounding box, allowing the area ratio of the cropped to original image to be 0.1 to 1.0, clipping boxes to the cropped image, and using a minimum overlap threshold of the cropped boxes to the new image to be 0.3, otherwise it is removed.

Lastly, we use Patch Gaussian to add noise to our data. This helps to make the model more robust, which is necessary for new images which could be blurry, at different scales, or even different locations (Lopes et al., 2019). It works by adding a patch of Gaussian noise, sized WxW, which allows the model to change between gaussian noise and Cutout (which removes all information in the patch). The patch size will be between 1 and 250 and the standard deviation of the noise within the patch will be between 0.0 and 1.0.

4. Experiments

The experiments were carried out using TensorFlow 2.0 Object Detection API, an open-source framework built on top of TensorFlow, the same as in (Beery et al., 2018). The Faster-RCNN with the ResNet backbone with the parameters outlined in the paper are used as our baseline model. This meant resizing images to a minimum dimension of 600 and a max dimension of 1024 and using SGD with a momentum of 0.9. We also use horizontal flipping for a data augmentation method and have an IoU threshold of 0.5. The paper (Beery et al., 2018) trained the baseline model with learning rate 0.0003 which decayed by a factor of 10 at $90k^{th}$ and again at $120k^{th}$ step until it reached a total of 200k steps. Unlike the original work, due to constraints of time and computing resources, our baseline model was only trained with $60k$ steps with starting learning rate at 0.0003, but decayed it using a cosine function from the $5k^{th}$ step. A cosine decay learning rate was chosen as it is a common and default policy for Faster-RCNN models in the API. The learning rate and steps were not optimized due to the same constraints but we argue that these parameters are less significant as our aim was to explore and compare different methods to improve generalisation. Instead of batch size 1 as used in (Beery et al., 2018), batch size of 4 is picked to speed up training as allowed by GPU. Using a range of further data augmentation methods we look at how domain generalisation can be improved. Our hyper-parameters and model settings are summarised in Table 2.

The evaluation metric being used is the mAp with IOU $\geq$ 0.5, where each image is evaluated individually and the specific frame number of the image in the snapshot sequence is disregarded. In (Beery et al., 2018), two methods of evaluation are performed, Most Confident and Oracle. The

Parameter	Chosen Value	Updated
Optimizer	SGD, momentum 0.9	
Batch Size	4	✓
# of Training Steps	60,000	✓
Learning Rate	$3e^{-4}$ w/ cosine decay	✓
Anchor Scales	0.25, 0.5, 1.0, 2.0	
Anchor Aspect Ratios	0.5, 1.0, 2.0	
Augmentation	Horizontal Flip Only	
Regularization	None	
Score Threshold	0.2	
IoU Threshold	0.5	

Table 2. Baseline Model Hyper-parameters

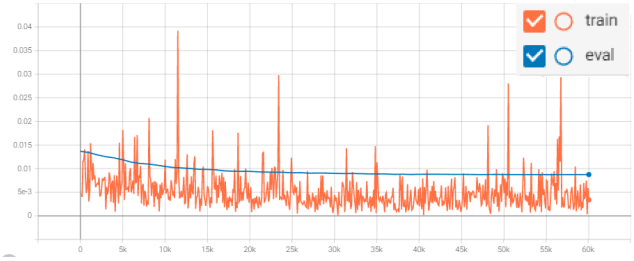

Figure 4. Example of Loss Function (Localization Loss) Plot in Experiments

accuracy in (Beery et al., 2018) is measured per sequence, where Most Confident returns a correctly labelled sequence if all images in the sequence correctly label the animal and have an IOU ≥ 0.5, whereas Oracle returns a correctly labelled sequence if at least one image in the sequence labels the animal correctly and has an IOU ≥ 0.5. Although utilising sequence data improves accuracy to a significant degree, it should not be important when solely testing for improvements to domain generalization as each image in the sequence will have the same background.

Using the TensorFlow API, all of the listed hyper-parameters for the two-staged model could be controlled using a configuration file. The aforementioned pre-trained model, namely "faster_rcnn_resnet101_v1_800x1333_coco17_gpu-8", could also be downloaded from the API and be specified in the configuration. The image dataset from (Beery et al., 2018) was converted into TFRecords using a Python script provided on the official API tutorial web-page. Following these instructions, the results given in this report are reproducible.

To improve model generalisation, we augmented the data and applied regularisation to the deep neural network. Figure 4 shows an example of changes to a loss function during training steps. The training loss fluctuates rapidly due to the small batch size and large size of the network. This disabled us from measuring the generalisation by comparing with training loss, e.g., calculating generalisation gap. Therefore, the generalisation performances are evaluated by comparing validation results with other models instead. Different methods were explored separately to provide a clear comparison. The experimental models and their accuracies and losses are shown in Table 3. All models were evaluated using the same validation set, i.e., the trans (unseen) dataset. Two baseline models, #1 and #2, were trained where #2 also contained cis dataset during training such that the amount of training data increased from 13,553 to 32,864 images. This roughly 150% increase in training data was able to improve the precision significantly from 40.2% to 47.1%, not to mention the extra cis dataset did not introduce new camera locations (#1 and #2 both looked at the images from the same cameras and backgrounds). This implies that simply gathering more training data is inevitably a good method to improve model performance on unseen data even though the new training data may contain the same biases. From model #2 onwards, models below were trained with the larger training set as in #2.

Among the data augmentation techniques, model #5 with random cropping and random coefficient 90% (10% of the images cropped) gave the best precision 49.9%. Model #3 with brightness adjustment performed the worst in precision but had the lowest objectness loss. This can be interpreted as adjusting brightness randomly may increase the variance of data, causing a lower loss, but lower accuracy. In the series of regularization, model #11 with L2 weight $1e^{-4}$ gave the best precision and recall among all trained models. The localization losses by RPN of all models were close and did not improve a lot using various methods. This is believed to be because the pre-trained model performed well enough in determining the boundaries of the bounding boxes.

Models #7 and #8 utilising dropout at 10% and 30% respectively, did not perform substantially better than the #2 baseline. Importantly, model #9 with 50% dropout was the only model that performed worse than the #2 baseline in precision (excluding the #1 baseline). This is likely because of the low training time. As the model is not exposed to the whole data for a significant number of iterations and does not exhibit any signs of overfitting, the dropout is likely negatively impacting the training of key features in the model instead of increasing domain generalization in any way.

Table 4 shows the experiment results using combined methods. Two models were trained where model #12 combined the modifications of models #3, #5, #6 and model #13 combined the models #5 and #11, the best in regularization. On top of these, model #13 combined model #12 and #11. Surprisingly, all of these three models did not further improve the accuracies. For model #12, the failure was believed to be caused by small step number such that the original and augmented images did not pass through the model for sufficiently enough epochs. Using 60k steps with batch size 4 equals to training 33k original images only with less than 8 epochs. There is a need to increase the step size especially when a number of augmentations are being adopted. Due to limited time and resources, there are also many more parameters to be configured but yet to be optimized.

	Model	mAp	Av. Recall	CL_C	LL_C	LL_{RPN}	OL_{RPN}
#1	Baseline	0.402	0.447	0.211	0.0404	0.0090	0.115
#2	Baseline (included cis dataset in training)	0.471	0.523	0.159	0.0378	0.0089	0.084
	Augmentations						
#3	Baseline #2 + Brightness Adjustment	0.475	0.500	0.132	0.0334	0.0092	***0.053***
#4	Baseline #2 + Random Cropping (75%)	0.483	*0.509*	*0.128*	***0.0323***	0.0091	0.108
#5	Baseline #2 + Random Cropping (90%)	*0.499*	0.508	0.136	0.0351	*0.0090*	0.078
#6	Baseline #2 + Patch Gaussian	0.481	0.507	0.154	0.0371	0.0091	0.105
	Regularizations						
#7	Baseline #2 + Dropout (10%)	0.478	0.506	0.151	0.0366	0.0089	0.075
#8	Baseline #2 + Dropout (30%)	0.489	0.514	0.134	0.0351	0.0091	0.073
#9	Baseline #2 + Dropout (50%)	0.462	0.482	0.141	0.0347	0.0092	0.065
#10	Baseline #2 + L1 regularization ($1e^{-4}$)	0.487	0.508	0.136	0.0341	0.0088	*0.061*
#11	Baseline #2 + L2 regularization ($1e^{-4}$)	***0.508***	***0.529***	***0.122***	*0.0335*	***0.0087***	0.069

Table 3. Experiment Results with Separate Methods (*italics* indicate the best results per series, **bold** indicate the best overall)

	Model	mAp	Av. Recall	CL_C	LL_C	LL_{RPN}	OL_{RPN}
#12	Baseline #2 + All Augmentation	0.483	0.510	0.152	0.0393	0.0090	0.079
#13	Baseline #2 + Random Cropping (90%) + L2 ($1e^{-4}$)	0.474	0.495	0.131	0.0332	0.0091	0.060
#14	Baseline #2 + All Augmentation + L2 ($1e^{-4}$)	0.472	0.483	0.132	0.0334	0.0091	0.043

Table 4. Experiment Results with Combined Methods

Figure 5. Model Results on Left and Answers on Right. (a) Correct Detection by Model #11 in Low Contrast Image; (b) False Positive Detection by Model #5 in Blurry Image Background; (c) False Negative Detection by Model #11 in Low Brightness Image

Figure 5 shows three examples of image detection using model #13 on the left side and the annotations on the right. The first image on the top shows a correct animal detection even though the image is grey and in low contrast. The second image gave a false positive detection on the blurry background, and the last image failed to detect object in an extremely dark environment. Patch Gaussian and brightness adjustment augmentations were expected to improve the second and third situations respectively.

Finally, the #1 and #2 baseline models and models #11, #12, #13 and #14 were tested on the test dataset. Model #11 was chosen as it was the best performing model on the validation set, and the rest were chosen to examine if the models with combinations of different data augmentations and regularizations would give better results when being evaluated on the much larger test set. These results were mainly inconclusive however, with no model being a stand out in terms of performance and none being much better than the #2 baseline model. These results, which can be seen in Table 5, present the idea that even though the validation and test datasets consist of images from locations not seen in training, that more new locations being present requires more robust model generalization than was achieved. This could be since the validation set only includes images from 1 location, but the test set includes images across 9 locations.

4.1. Discussion

Our results from Table 3 and Table 5 highlight that an increase in training data is by far and away the best way to improve on the performance of the baseline model. This is likely emphasised due to the limited number of certain classes in the original training dataset. Although the performance of the #2 baseline model could likely be increased substantially further by simply adding more training data or using images of a higher resolution instead of the downscaled benchmark images, this would eventually start to give diminishing returns. This is due to the validation and test data being comprised of images from locations not included in the training data.

When examining our validation results overall, out of the data augmentations applied, random cropping (90%) was the best. Out of the regularization techniques applied, L2 regularization ($1e^{-4}$) was the best. Somewhat worryingly however, was that combining these two techniques led to a noticeable drop in performance, even slightly worse than the #2 baseline performance in average recall. The main conclusions to be made from our validation results are that data augmentation and regularization can help a model generalize to data distributions not seen in training, but certain techniques and combinations of techniques can degrade model performance if being applied on a limited dataset and/ or not a large number of epochs during training.

Somewhat inconclusively however, when examining the test set we see little difference between the performance of each model. As mentioned earlier, this is likely due to the test set containing images from 9 new locations, unlike the validation set which only contains images from 1 new location. This could be combated by possibly introducing harsher regularization techniques, or simply by increasing the hyper-parameter value of L2 regularization, for example.

5. Related work

There have been numerous other works looking at how to detect animals in camera trap images. Domain generalisation is a large part of camera trap models, in that models working on cameras set up at new locations need to be able to work as well as locations that the model has been trained on. There has been work looking at how to improve this kind of generalisation (Sagawa et al., 2019; Beery et al., 2020).

(Sagawa et al., 2019) look at minimising the worst case training loss on a set of classes and argue that poor generalisation across certain classes has a big effect on performance. This could be due to imbalanced classes and lack of training data for some classes. They come up with Distributionally Robust Optimisation with stronger regularisation to improve worst case accuracy. (Beery et al., 2020) use synthetic classes in the Caltech Camera Traps dataset to improve generalisation for rare classes. Classes that appear rarely in the data are added to with simulated data, looking at different kinds of variation such as pose, lighting, model, and simulation method. They found the more they increased this synthetic data and the more variation they added, the more the accuracy on these rare classes improved.

Some works have looked at improving object detection on datasets with few labels. Active learning has been implemented to reduce the amount of work humans need to put into annotations, and only input when the model needs help (Norouzzadeh et al., 2021). They utilise transfer learning and Active Learning to identify and count species. (Norouzzadeh et al., 2021) use this on the Snapshot Serengeti dataset and the NACTI dataset to find the most significant images in which humans need to label. They use Faster R-CNN on the images and then used the cropped images produced by this model with a pre-trained embedding model. They then use active learning to help fine tune the classification model. There are examples of experiments that split the task into species classification and empty-image classification (Tabak et al., 2020), as a means to tackle the largely imbalanced dataset. Transfer learning is often used on smaller datasets that would otherwise not have enough data to train effectively on these datasets (Schneider et al., 2018; Norouzzadeh et al., 2021).

(Tabak et al., 2020) use 3 million Camera Trap images from 18 studies in 10 states across the USA. They use this to train two models, one for species classification and empty-image classification. This can be helpful to remove empty images, and train on only images with animals. However, they still argued transferability across locations is an issue.

	Model	mAp	Av. Recall	CL_C	LL_C	LL_{RPN}	OL_{RPN}
#1	Baseline #1	0.438	0.399	0.205	0.0413	0.0261	0.134
#2	Baseline #2	0.483	0.424	0.128	0.0374	0.0259	0.104
#11	Baseline #2 + L2 regularization ($1e^{-4}$)	0.486	0.419	0.123	0.0365	0.0261	0.091
#12	Baseline #2 + All Augmentation	0.484	0.420	0.135	0.0378	0.0243	0.095
#13	Baseline #2 + Random Cropping (90%) + L2 ($1e^{-4}$)	0.485	0.421	0.123	0.0333	0.0265	0.081
#14	Baseline #2 + All Augmentation + L2 ($1e^{-4}$)	0.485	0.419	0.116	0.0333	0.0243	0.060

Table 5. Test Results

(Yousif et al., 2017) use joint background modelling and deep learning classification to detect humans and animals in cluttered images, reducing complexity of the DCNN.

Some works have used attention mechanisms in object detection models in the aim of modelling the domain specific weights, to then alter the feature weights for generalization (He et al., 2018). This work uses a SSD (Single Shot MultiBox Detector) model that is reported to be 3 times faster than Faster-RCNN (Liu et al., 2016). When testing on a test set containing images from 4 scenes that were excluded from a training set of 12 scenes, the model achieves better performance than the other methods examined. Although, in certain situations without domain labels, the attention block can provide weights that reduce the model performance.

6. Conclusion

In this paper we have explored different domain generalisation techniques in order to try and improve upon performance in camera trap datasets. We have used Faster R-CNN as a state-of-the-art model and looked at different data augmentation techniques, as well as different regularisation methods. Our results on the validation set show that despite the data augmentation techniques performing better than the baseline, L2 regularisation actually obtained the best mAp score of 0.508. We also found that random cropping performed better than all the other data augmentation techniques, as well as all data augmentations combined. Brightness adjustment was not as effective in this setting as hoped. On the test set, there was much less increase in the performance as anticipated. The results in this paper suggest that in order to improve domain generalisation on a fixed dataset, other methods should be looked at. If there is potential for more data/ a more balanced dataset, this should be favoured as the difference in baseline #1 and baseline #2 shows a 17% increase in mAp score.

6.1. Future works

In the future, it would be useful to look at how changing the probabilities that a frame is augmented for each type of data augmentation can alter the performance of the model. This could be done by looking at mAp scores at different probabilities. If we are looking at simply increasing performance for a specific model, different IoU thresholds could be looked at, and precision recall curves created for each threshold. Using these curves, we could optimise the threshold. It could also be useful to add synthetic data to classes with few examples so that the model can learn more about the representations of these animals. For example, GANs could potentially be used in order to generate more data without the need for going out and collecting more data. This could help to create a more balanced and useful dataset.

A. Code - GITHUB link

https://github.com/derekywk/mlpg037

References

Beery, Sara, Van Horn, Grant, and Perona, Pietro. Recognition in terra incognita. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 456–473, 2018.

Beery, Sara, Liu, Yang, Morris, Dan, Piavis, Jim, Kapoor, Ashish, Joshi, Neel, Meister, Markus, and Perona, Pietro. Synthetic examples improve generalization for rare classes. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 863–873, 2020.

Burton, A Cole, Neilson, Eric, Moreira, Dario, Ladle, Andrew, Steenweg, Robin, Fisher, Jason T, Bayne, Erin, and Boutin, Stan. Wildlife camera trapping: a review and recommendations for linking surveys to ecological processes. *Journal of Applied Ecology*, 52(3):675–685, 2015.

He, Weixiong, Zheng, Huicheng, and Lai, Jianhuang. Domain attention model for domain generalization in object detection. In Lai, Jian-Huang, Liu, Cheng-Lin, Chen, Xilin, Zhou, Jie, Tan, Tieniu, Zheng, Nanning, and Zha, Hongbin (eds.), *Pattern Recognition and Computer Vision*, pp. 27–39, Cham, 2018. Springer International Publishing. ISBN 978-3-030-03341-5.

Liu, Wei, Anguelov, Dragomir, Erhan, Dumitru, Szegedy, Christian, Reed, Scott, Fu, Cheng-Yang, and Berg, Alexander C. Ssd: Single shot multibox detector. In Leibe, Bastian, Matas, Jiri, Sebe, Nicu, and Welling, Max (eds.), *Computer Vision – ECCV 2016*, pp. 21–37, Cham, 2016. Springer International Publishing.

Lopes, Raphael Gontijo, Yin, Dong, Poole, Ben, Gilmer, Justin, and Cubuk, Ekin D. Improving Robustness With-

out Sacrificing Accuracy with Patch Gaussian Augmentation. 6 2019. doi: 10.48550/arxiv.1906.02611. URL https://arxiv.org/abs/1906.02611v1.

Norouzzadeh, Mohammad Sadegh, Morris, Dan, Beery, Sara, Joshi, Neel, Jojic, Nebojsa, and Clune, Jeff. A deep active learning system for species identification and counting in camera trap images. *Methods in Ecology and Evolution*, 12(1):150–161, 1 2021. ISSN 2041210X. doi: 10.1111/2041-210X.13504.

Ren, Shaoqing, He, Kaiming, Girshick, Ross, and Sun, Jian. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, 6 2015. ISSN 01628828. doi: 10.48550/arxiv.1506.01497. URL https://arxiv.org/abs/1506.01497v3.

Sagawa, Shiori, Koh, Pang Wei, Hashimoto, Tatsunori B, and Liang, Percy. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. *arXiv preprint arXiv:1911.08731*, 2019.

Schneider, Stefan, Taylor, Graham W, and Kremer, Stefan. Deep learning object detection methods for ecological camera trap data. In *2018 15th Conference on computer and robot vision (CRV)*, pp. 321–328. IEEE, 2018.

Schneider, Stefan, Taylor, Graham W., Linquist, Stefan, and Kremer, Stefan C. Past, present and future approaches using computer vision for animal re-identification from camera trap data. *Methods in Ecology and Evolution*, 10 (4):461–470, 4 2019. ISSN 2041210X. doi: 10.1111/2041-210X.13133.

Shorten, Connor and Khoshgoftaar, Taghi M. A survey on Image Data Augmentation for Deep Learning. *Journal of Big Data*, 6(1):1–48, 12 2019. ISSN 21961115. doi: 10.1186/S40537-019-0197-0/FIGURES/33. URL https://journalofbigdata.springeropen.com/articles/10.1186/s40537-019-0197-0.

Song, Pinhao, Dai, Linhui, Yuan, Peipei, Liu, Hong, and Ding, Runwei. Achieving Domain Generalization in Underwater Object Detection by Image Stylization and Domain Mixup. URL https://github.com/mousecpn.

Tabak, Michael A., Norouzzadeh, Mohammad S., Wolfson, David W., Newton, Erica J., Boughton, Raoul K., Ivan, Jacob S., Odell, Eric A., Newkirk, Eric S., Conrey, Reesa Y., Stenglein, Jennifer, Iannarilli, Fabiola, Erb, John, Brook, Ryan K., Davis, Amy J., Lewis, Jesse, Walsh, Daniel P., Beasley, James C., VerCauteren, Kurt C., Clune, Jeff, and Miller, Ryan S. Improving the accessibility and transferability of machine learning algorithms for identification of animals in camera trap images: MLWIC2. *Ecology and Evolution*, 10(19):10374–10383, 10 2020. ISSN 20457758. doi: 10.1002/ECE3.6692.

Touh, Panawè and Sharma, Mukesh. Detection and Counting of Animals in Camera-Trap Images using Faster R-Cnn and Data Augmentation Techniques. *International Journal of Engineering and Advanced Technology*, 2020. doi: 10.35940/ijeat.E9925.069520. URL www.ijeat.org.

Vecvanags, Alekss, Aktas, Kadir, Pavlovs, Ilja, Avots, Egils, Filipovs, Jevgenijs, Brauns, Agris, Done, Gundega, Jakovels, Dainis, and Anbarjafari, Gholamreza. Ungulate Detection and Species Classification from Camera Trap Images Using RetinaNet and Faster R-CNN. *Entropy (Basel, Switzerland)*, 24(3):353, 2 2022. ISSN 1099-4300. doi: 10.3390/E24030353. URL https://pubmed.ncbi.nlm.nih.gov/35327863/https://pubmed.ncbi.nlm.nih.gov/35327863/?utm_source=FeedFetcher&utm_medium=rss&utm_campaign=None&utm_content=1fK6RBI2YOGNyXF39xDqh1a-ff8j6eCV2pzN5YxrnnXf40fQVf&fc=None&ff=20220325072458&v=2.17.6.

Yousif, Hayder, Yuan, Jianhe, Kays, Roland, and He, Zhihai. Fast human-animal detection from highly cluttered camera-trap images using joint background modeling and deep learning classification. *Proceedings - IEEE International Symposium on Circuits and Systems*, 9 2017. ISSN 02714310. doi: 10.1109/ISCAS.2017.8050762.