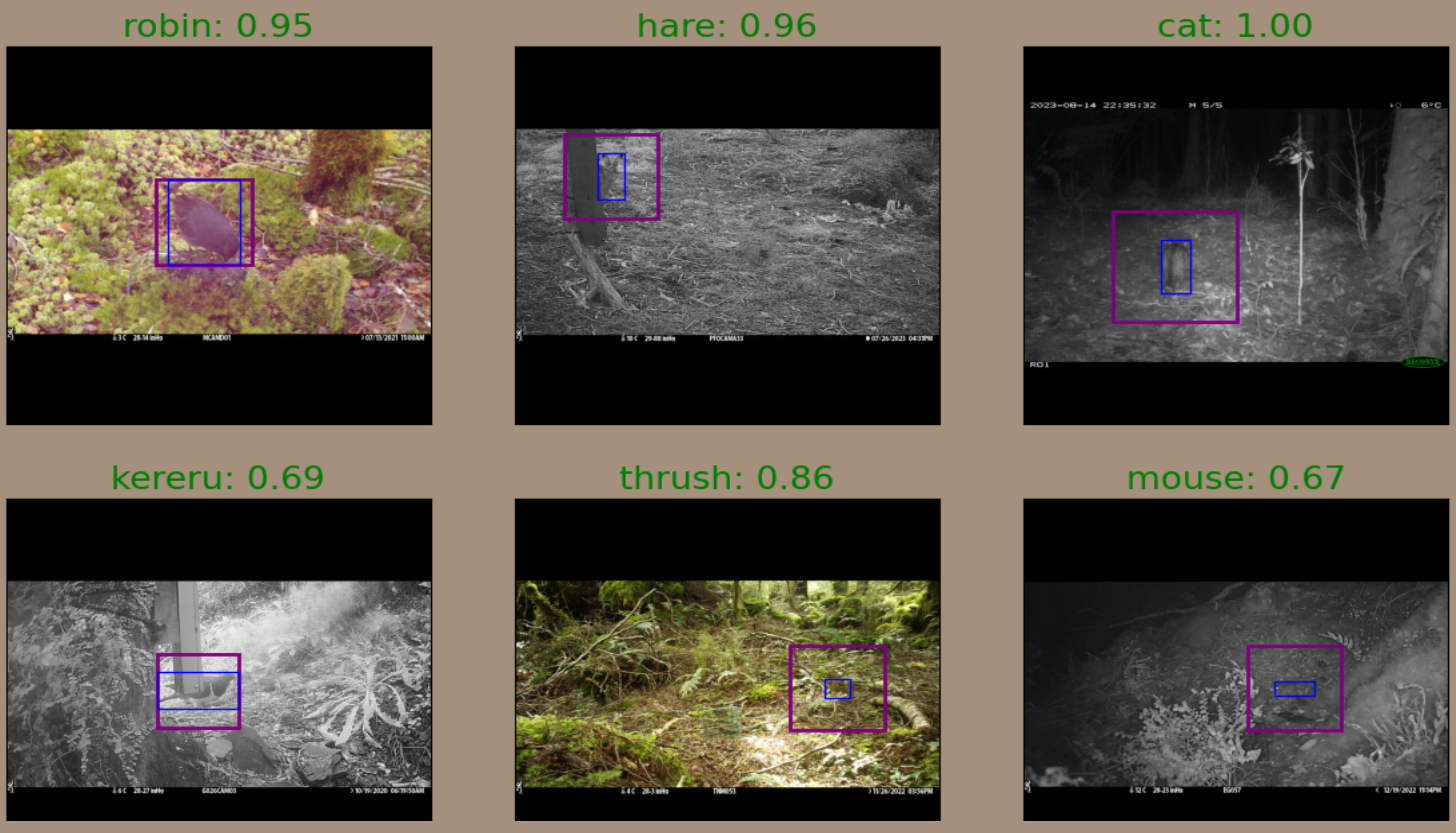
Author: Olly Powell
Prepared for: The Department of Conservation
Project: DOC-7729484/5
Period: January - August 2025

wekaResearch

2025 Image Classification Updates

Summarising improvements to DOC's camera trap computer vision model *Alita* up to Version 3.03, and the associated image dataset.

Report: DOC-10447968 **Revision:** 03 October 6, 2025



SUMMARY

Substantial improvements were made to the image datasets, data processing pipeline and model training algorithms. The resulting performance change is summarised in the table 1. These results were produced by running each model over test images from study areas not used for training.

TABLE 1: PERFORMANCE CHANGE SINCE VERSION 2.01

Version	Released	Balanced Acc	mAP	Classes	Train Images	Test Images
2.01	Nov 24	0.69	0.77	78	446,760	6553
3.01	Jul 25	0.90	0.94	78	376,486	6553

Other highlights from this work include:

- Adoption of a smaller neural network architecture whilst fine-tuning four additional layers. This boosted accuracy and speed whilst reducing memory use.
- Development of new sub-sampling methods to prioritise images by their uniqueness. This assisted with balancing the training dataset whilst retaining diversity.
- Inclusion of additional training data for weaker classes, and three new classes: Fernbird, long-tailed cuckoo and kākāpō
- Changing from multi-class to multi-label prediction also enabled the combined logic from the detector and classifier models to reduce mistakes on empty images, and flag 'unknown' images rather than creating false positives.
- The change from multi-class to multi-label prediction also resulted in predictions from individual species being independent of each other. In principle this should make the model more consistent for the population monitoring use case, without hurting recall for incursion response and surveillance.
- Packaging with PyInstaller and creation of a simple graphical user interface.
- Integration into the AddaxAI platform.

Contents

1 INTRODUCTION

1.1 GLOBAL CONTEXT

DOC began working with wekaResearch on image classification in 2022. At this time Microsoft's *AI for Good lab* had released version 5 of *MegaDetector* (MDv5) [1], based on the YOLOv5 repository developed by Glenn Jocher at Ultralytics [2]. MDv5 is an object detection model that works with camera trap images and predicts bounding boxes around just three classes: *Animal*, *Human*, and *Vehicle*.

MDv5 was trained from a very large collection of training images, few of which are sourced from New Zealand. The strength of this model is that it produces an animal detector that generalises well, and can identify if there is anything interesting to us in an image and where in the image it is. MegaDetector has since been widely adopted for a variety of conservation purposes globally.

In 2023 the MegaDetector repository was forked (copied and continued independently) by one of its authors, Dan Morris. Dan has continued maintenance and development of that fork [3] along with the dataset repository LILA BC. Meanwhile, development of the original repository has continued by the Microsoft team and re-named *PyTorch Wildlife*.

In conjunction with the above, the visualisation tool *Timelapse* was developed at the University of Calgary to enable more convenient user filtering and verification of the image predictions. The data formats for Timelapse and MegaDetector are compatible, and have since been widely adopted.

The number of alternative camera trap classification models globally has proliferated in recent years, and many are using compatible JavaScript Object Notation (.json) format of MegaDetector and Timelapse.

More recently Peter Van Lunteren developed a platform AddaxAI that runs a variety of models incorporating the same data format, integrating Timelapse's visualisation methods into the same platform. DOC has collaborated with Peter to add two models relevant to the New Zealand context into AddaxAI.

1.2 PROJECT HISTORY

From the start, our work has adopted a two-stage approach. MegaDetector to localise any points of interest within a large image (and potentially eliminate empty images), followed by our own methods for fine-grained classification.

In the beginning it wasn't obvious that the two-stage approach was justified. A simpler path would have been to freeze the lower layers of MDv5 and re-train only the clas-

sification head. But that would have left our efforts somewhat dependent on MegaDetector and underlying YOLOv5 architectures.

By keeping a completely separate detection and classification step, we have the flexibility to try different detection models.

The second reason for the two-stage approach is performance. By extracting fixed size crops, we work efficiently with a small image size but with no loss of detail. This advantage was confirmed in our own work, and independently by the Google Research team with the release of *SpeciesNet*, which adopted the same concept [4], as have others.

1.2.1 ALITA VERSION 1

Version 1 ran MegaDetector in a stand-alone Python environment. A separate classification pipeline was built based on extracting 480×480 crops. The crops were passed to a multi-class classifier, with 52 classes.

The classifier was trained on 250,000 pre-cropped images. All original images were acquired from the Doubtful valley in the Lewis Pass. Validation and test datasets consisted of 10% each randomly chosen from the same pool.

Balanced accuracy of 0.91 was reported [5], but this was misleading as there were likely near-duplicates spread between the three splits.

1.2.2 ALITA VERSION 2

Version 2 was trained on 2.5 million images from 50 study areas, with up to 95 cameras per study area. The full image dataset was uploaded to LILA BC so that other groups can develop competing models.

Validation was performed with randomly chosen images from the same pool as training, whilst test split was from 5 study areas set aside from the training data for that purpose.

Experiments were made to balance the training data and develop a more effective image augmentation pipeline in order to improve generalisability. Balanced accuracy on the test set was reported to be 0.78 over 87 classes [6].

Training, inference and setup of the miniConda environment were all run from the terminal through PowerShell scripts. The process could be run on video as well as still images, though performance on video was notably less accurate.

2 ALITA VERSION 3

2.1 MULTI-CLASS TO MULTI-LABEL

In earlier versions of Alita, the classification of the cropped image was treated as a multi-class problem. Conceptually this means we assume one class is present, and the challenge is to identify it. Mathematically the final activation step applied to each class output z_i is the *softmax*, shown in equation 1.

$$p_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{1}$$

The sum over all probability scores p_i is inherently equal to 1 and the values of z_j from other classes affect the probability score of class *i*. By contrast the activation function for our multi-label classifier is a *sigmoid* function, shown in equation 2.

$$p_i = \frac{1}{1 - e^{z_i}} \tag{2}$$

In equation 2 the probability p_i score for class *i* has no dependence on the score z_j from any of the other classes. The practical importance of this change is two-fold:

1. We do not need a specific class for identification of empty images. Empty can be simply inferred from the absence of any known classes above some threshold.

2. We can still choose to use the maximum prediction, if distinguishing between species is the most important goal Otherwise we can look at the species of interest along with a suitable threshold and ignore the other columns. In the latter case, our prediction scores are unaffected by any likeness in the image to other species.

2.2 OUTPUT FORMATS

Running the Alita model through the GUI produces at least two files.

1. A .json file with bounding boxes, and the class with the highest prediction score for that image.

2. A .csv file with the top three scoring classes. One column is the 'encounter' prediction, by which we mean the highest scoring class from a burst of images, or a single video file. Individual scores for all 81 classes are also provided.

It is difficult to derive statistics from the .json file directly. The intention of this file is to allow the user to verify the results visually and confirm suitable thresholds in Timelapse. Timelapse has filtering options to allow the user to visualise results conveniently.

The .csv file is intended for more advanced use cases such as population monitoring, where the user is likely to use the scores directly for their own statistical methods in R or Python.

2.2.1 SPECIES OF INTEREST

There is also an option in the GUI for the user to select particular 'species of interest'. In doing this an additional .json file is provided, where all animals are assumed to be from that class. This allows the user to experiment with binary thresholds for a single class at a time, whilst instantly visualising the effect in Timelapse.

No change is made to the model or scores by selecting this option.

2.3 AGGREGATION AND ENSEMBLING LOGIC

The 'encounter' prediction in one column of the .csv file is an aggregation of rows where the timestamp from each image is within 30 seconds of the last image, or all the frames that come from a single video.

The encounter prediction is also an ensemble from the two models, Alita and Megadetector. The possible outcomes for the encounter are listed below.

1. Any of 81 available species, all the prediction scores are provided in the CSV.
2. **Or** *Empty*, where **Both** models were below their respective thresholds.
3. **OR** *Unknown*, where the MegaDetector predicted an animal over 0.15, but Alita provided no scores over the classifier threshold, which is selectable by the user in the drop-down menu.

This logic can be seen in action in figure 1. In this example eight of the nine images are correct with good scores.

The second image in this example contains a cat, where MegaDetector has placed an accurate box (in blue) around the animal, and scored some value over its threshold of 0.15. However, the maximum score from Alita (presumably for a cat) came to 0.49, falling below the binary threshold of 0.5, resulting in an *unknown* label.

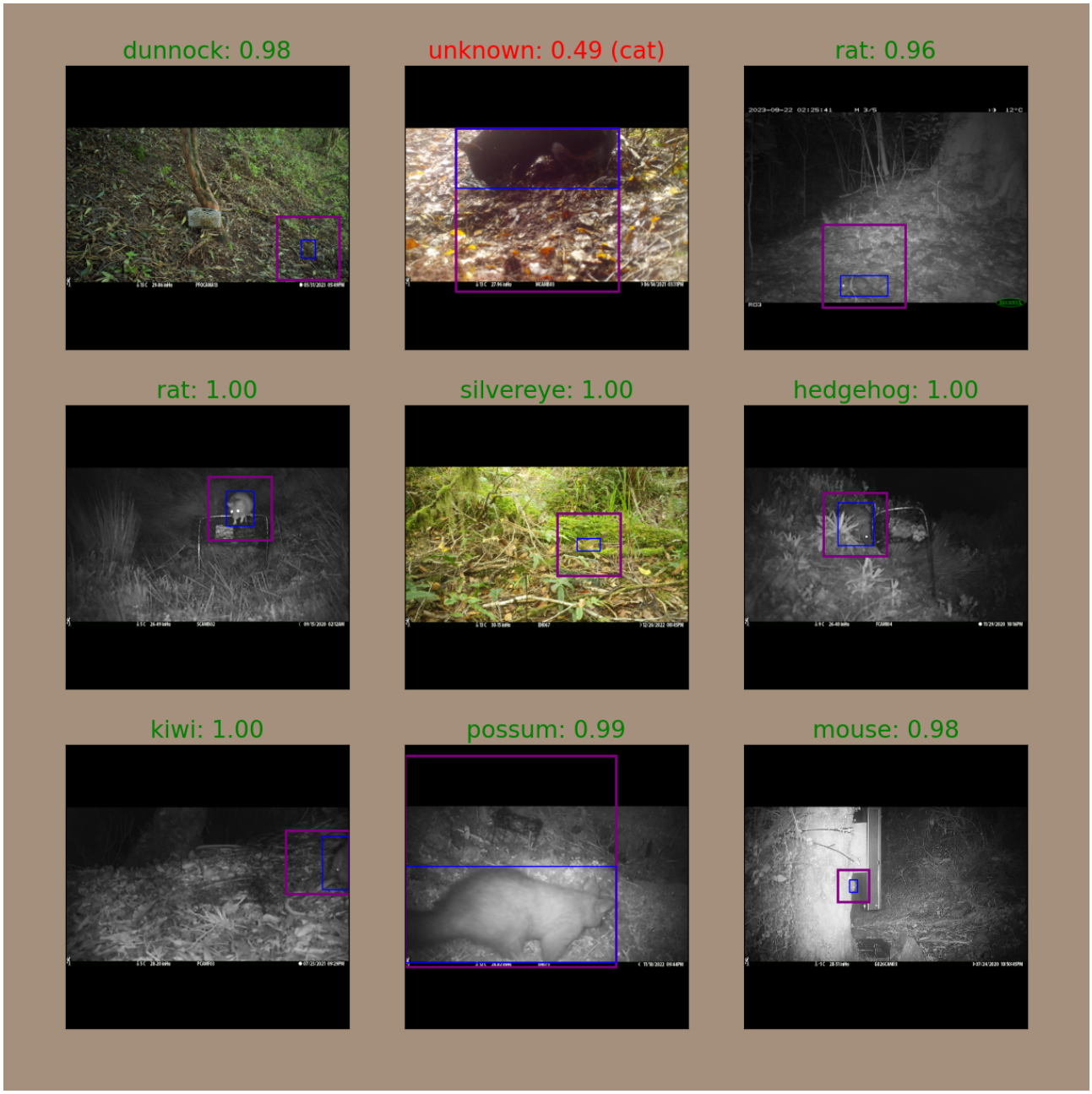


FIGURE 1: A SELECTION OF IMAGES FROM THE VALIDATION SPLIT. MEGADETECTOR BOUNDING BOX IS SHOWN IN BLUE, AND THE CROP FOR ALITA SHOWN IN PURPLE. THE GREEN LABEL IS A CLASS PREDICTION WITH ITS CONFIDENCE SCORE.

2.4 NETWORK ARCHITECTURE

Alita was trained with transfer learning applied to the EfficientNet family of neural network backbones.

The full network consists of the pre-trained backbone and a randomly weighted classification head of a custom design. Initially only the classification head is trained. As improvements diminish, layers of the backbone can be progressively unfrozen for fine-tuning.

Alita V2 was based on EfficientNetV2L, fine-tuning only the head and the top two backbone layers. Smaller networks were tried at the time but with some loss of performance.

In the experimentation for V3 several smaller backbones were tried whilst tuning up to 12 backbone layers. It was found that tuning 6 layers provided a step change in performance over 4 or 2. Beyond 6 no further gains were observed.

With 6 layers tuned, there was negligible dependence on network size within the range trialled. The results from these experiments are summarised in table 2.

TABLE 2: PERFORMANCE WITH VARIOUS NETWORK SIZES AND LAYERS TUNED

Backbone	Exp-Run	Params (M)	Lyrs	Tuned BB Lyrs	Batch Size	B-Acc	mAP
B0	46-10	4.1	82	12	16	0.88	0.94
B0	46-12	4.1	82	8	16	0.87	0.94
B0	46-13	4.1	82	6	16	0.89	0.95
B0	46-14	4.1	82	4	16	0.61	0.68
B2	46-22	7.8	91	6	32	0.83	0.92
B3	46-18	10.8	97	6	16	0.85	0.91
V2S	46-21	21.0	137	6	32	0.89	0.94
B5	46-11	28.5	117	2	16	0.66	0.76
V2M	46-20	54.0	208	6	16	0.69	0.77
V2L	39-01	120	309	2	16	0.69	0.77

For the remainder of this work, EfficientNet-V2S was used, whilst fine-tuning six backbone layers in addition to the classifier head. With 24 Gb of VRAM this allows us to train with a batch size of 32.

2.5 DATA

Additional labelled data was included in the overall dataset, bringing the total to approximately 3.5 million images. The distribution of class sizes is shown in Appendix B, Figure 7. Seven additional study areas were added for the out of domain test set, taking the total to 13.

Ideally the source data would be split so that each of training, validation and test splits would come from unrelated study areas, with each split containing every class. In practice the limited number of camera locations, extreme class imbalance and likely correlation between images from a given camera, make this impossible for now.

As a compromise, the schema summarised in table 3, was used for data splits. The validation and test metrics have different strengths and weaknesses. The test set is closer to what might be considered real-world performance, as the images were from different times of year or study areas. Only class sizes of at least 30 images were used for reporting metrics. The validation set contains most of the classes, but is less independent, as images can come from cameras nearby to those that provided images for training.

TABLE 3: SCHEMA FOR SPLITTING SCHEME FOR TRAINING IMAGES

Split	Images	Rules
Train	380,000	Class numbers limited by discarding per region and camera
Validation	40,000	Random choice of cameras, not used in training
Test	6,500	13 specific (unused above) regions, near-duplicates removed

One could reasonably question the importance of reaching an ideal test split, where having every class well represented. For end-users, it would be helpful to know the expected precision and recall for each class. However, if we are confident of consistent behaviour between classes then having every class well represented in the test images may be unnecessary.

2.6 DATA SUB-SAMPLING

Only a subset of the full image dataset was used for training. This was done to reduce class imbalance, and also to discourage the model learning irrelevant context from camera locations.

The baseline approach established in version 2 was to randomly discard images by class to limit the maximum occurrence of that class per camera and per study area. Limits of 200 and 2000 respectively were found by experiment.

In this update, new methods were tried with the goal of achieving the same balance but selecting the images with maximum diversity. These new methods are described next.

2.6.1 PERCEPTUAL HASHING

Hashing is a practice where a fixed function is applied to an image to reduce it to an easily stored string of values representing features of the image, referred to as a *hash*. Any two hashes can be compared efficiently by calculating their Hamming distance.

There are a variety of different hashing kernels, here we used perceptual hashing (pHash) from the ImageHash Python library. pHash combines size & depth reduction with discrete cosine transforms. This provides a robust way to represent *low-level* features that make up the structure of the image, such as relative sizes and positions.

The nature of our task is to identify small animals in otherwise static backgrounds. If used as-is, any pair of images from the same camera would be problematically similar. The solution proposed here was to use object detection first to crop the image with bounding boxes. The remaining crop plus a small buffer is mostly animal, so more suitable for meaningful hash comparison.

2.6.2 EMBEDDING DISTANCE

By running an image through a pre-trained neural network backbone it can be represented by an N-dimensional feature vector. By projecting these vectors onto an N-dimensional hypersphere, we can use the cosine of the angle between them as a metric for similarity.

The incentive to use embeddings as described above is that we are looking at the images the same way a network does. Network backbones can be pre-trained to recognise *high-level* features that are relevant to our task, such as the texture of fur, or feathers or the shape of an animal's head.

With the embedding distance approach implemented here, an upper limit to images was set per camera-class or region-class. Images were chosen by selecting the first at random, then subsequent images were chosen based on being furthest from those already chosen. The process continues until the upper limit for that class was reached. In this manner we can efficiently choose a small group of maximally different images from a large pool.

2.6.3 SUB-SAMPLING EXPERIMENTS

Experiments were run with sub-sampling based on combinations of perceptual hashing and embedding distance. The results are shown in table 4. The best performance was

achieved in experiment 53, which used a combination hashing first, on a per-camera basis, followed by embedding distance both per-camera and per study area.

TABLE 4: PERFORMANCE WITH VARIOUS COMBINATIONS OF PERCEPTUAL HASHING AND EMBEDDING BASED LIMITS PER CLASS BY CAMERA AND BY STUDY AREA

Exp-Run	Min H Dist.	Cam. Lim.	Area Lim.	Method	Images	B-Acc	mAP
46-09	-	200	3000	Rand.	351,094	0.887	0.944
49-01	50	180	2000	E-dist.	349,794	0.890	0.945
50	50	160	2000	E-dist.	341,164	0.905	0.955
51	-	160	2000	E-dist.	349,521	0.895	0.941
52	-	140	2000	E-dist.	342,955	0.876	0.940
53-02	60	160	2000	E-dist.	339,349	0.918	0.960
54	60	150	2000	E-dist.	335,211	0.911	0.953
55	70	160	2000	E-dist.	335,706	0.890	0.942

Comparing experiment 53 with experiments 49 and 50 suggests that the hashing process is performing a useful role.

Comparing 53 with 54 and 49 suggests that the embedding limit approach is also doing something useful, but there is some optimum near 160 images per class per camera for this.

The new methods appear to have accomplished a small performance boost overall of 2-3 percentage points from the previously optimised random discard strategy. That this is accomplished with fewer samples is quite encouraging.

2.7 INTEGRATION OF MEGADETECTOR

Initially PyTorch Wildlife (PW) was incorporated as we thought that would be the more regularly updated repository. However, PW was later commented out and replaced with the original MegaDetector as neither the new PW models nor the error handling in the framework methods appeared to be reliable yet.

Previous versions used a completely separate Python environment in order to avoid any package conflicts. The environments were merged to simplify packaging for version 3.

Some small modifications were made to the MegaDetector inference code, to enable it to run in the same environment as Alita. Those scripts have been re-named so that they do not get accidentally over-written in future updates.

2.8 PACKAGING AND GRAPHICAL USER INTERFACE

A simple Graphical User Interface (GUI) was made based on PyQT. A screenshot of the GUI is shown in Appendix A. Python and all the required dependencies were packaged into an .exe file using PyInstaller. Any images, weights and style files are stored in a single folder.

This approach results in an application that takes approximately one minute to unpack each time it is used. It is relatively easy to maintain and re-package, and does not require the user to make any permanent changes to their operating system.

3 CURRENT PERFORMANCE

The performance reported here is based on the most recent experiment (Exp_60), including the most update training data available at the time. With the increased data, three additional classes were included: Fernbird, long-tailed cuckoo and kākāpō, taking the total to 81.

The balanced accuracy over the 29 classes adequately represented in the out-of domain test split was 0.92. Precision and recall scores for individual classes are provided in Appendix E.

3.1 PREDICTION SEPARATION

A perfect classifier would score all true positives at close to 1 and true negatives close to 0. Under this ideal scenario, predictions of presence or absence of a class should be practically invariant to the choice of binary threshold, or minor changes to image capture conditions or locations.

We can visualise the degree to which this is being accomplished by plotting the true negative scores in a different histogram to the true positives. This is shown below in figure 2 for all classes from the validation dataset.

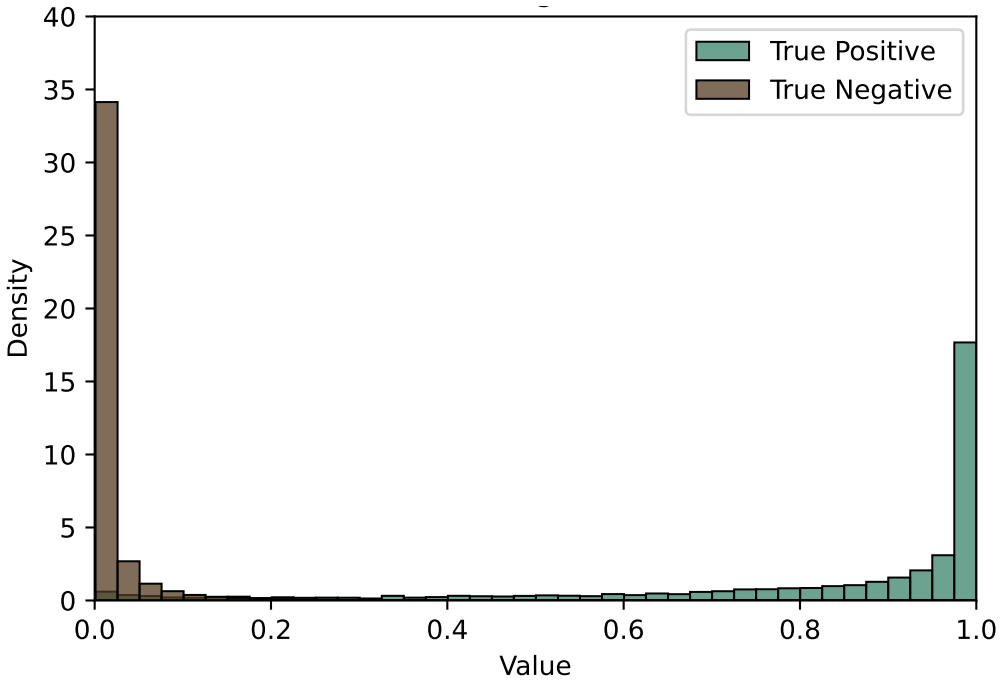


FIGURE 2: HISTOGRAMS FOR TRUE POSITIVES AND TRUE NEGATIVES FROM THE VALIDATION IMAGES ACROSS ALL CLASSES

We can see from figure 2 that we have excellent separation for the majority of cases, with just a small sliver of overlap. The mean score for true positives was 0.82, whilst the

mean for true negatives was 0.02. This is a great improvement over previous versions of Alita. A binary threshold between 0.3 and 0.6 would work fine for these images.

In future, it would be good to capture this information in a metric to monitor incremental improvement between models. We ought to consider metrics such as Cohen's D scores, that compare the mean true positive with mean true negative scores and normalise with some measure of spread.

3.2 AVERAGE PRECISION SCORES

The average precision (AP) scores give a convenient number to assess model quality with a single value per class. The AP scores from the test split are shown in figure 3.

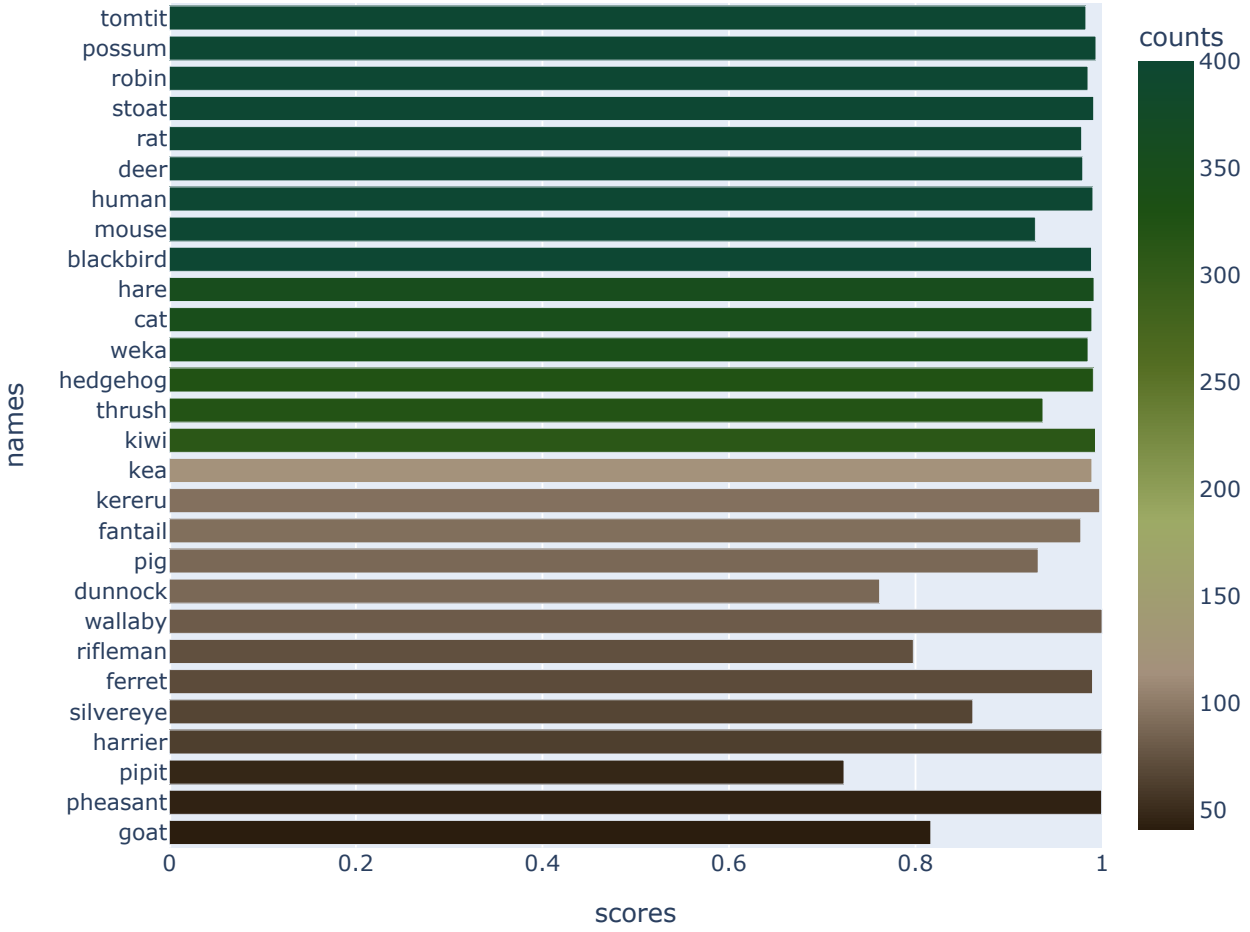


FIGURE 3: AVERAGE PRECISION SCORES BY CLASS FOR THE SET IMAGES

Over all classes Alita has a Macro AP (MAP) score of 0.96, with consistently high individual AP scores over all the classes we have adequate data. The scores for the whole validation split (which contains all the classes) is given in Appendix D.

3.3 CONFUSION MATRIX

3.3.1 CONFUSION ON VALIDATION IMAGES

Strictly speaking, for a multi-label classifier, each of the 81 species has its own confusion matrix (True Positive, False Positive, True Negative, False Negative). However, with such high precision scores there is little to be gained by looking at those individually.

We can gain some insight by considering the outputs as multi-class, and examining the highest scoring class for each image. The normalised confusion matrix, shown in figure 4 was constructed using the maximum score across all classes present in the validation split. This is similar to using the 'encounter' column in the .csv output, though without the additional aggregation logic.

There were five points here that stand out from the diagonal line as false positives for some rarer class. The target class on the horizontal axis (Bellbird, Dog, Fantail, Magpie, Rifleman) was in the image, but the predictions were other classes (grey warbler, black billed gull, black backed gull, spurwing plover, grey warbler again) respectively. In most cases these were based on very small sample sizes for the false class. Whilst this is undesirable, it is not indicative of a trend.

More notable was shore plover and Canada goose scoring numerous false positives resulting in a discernible horizontal row faint markers. This is most likely due to low confidence on these classes. This suggests these are easily out-scored by other classes. In both cases we have relatively few training images for these classes and would likely benefit from more training samples.

3.3.2 CONFUSION ON TEST IMAGES WITH THE ENCOUNTER OUTPUT

The introduction of the 'unknown' class means that if we were running Alita in real use, many of the mislabelled images previously discussed from figure 4 would likely have been labelled 'unknown' or sometimes 'empty' rather than create a false positive, as they would have scored below the binary threshold.

The confusion matrix in figure 5, was created for the test dataset, including empty images. Here the additional aggregation logic that makes up the 'encounter' prediction was included. Now we have virtually no false positives, but a faint line of mistakes that have been labelled 'unknown', as their maximum scores were below the binary threshold.

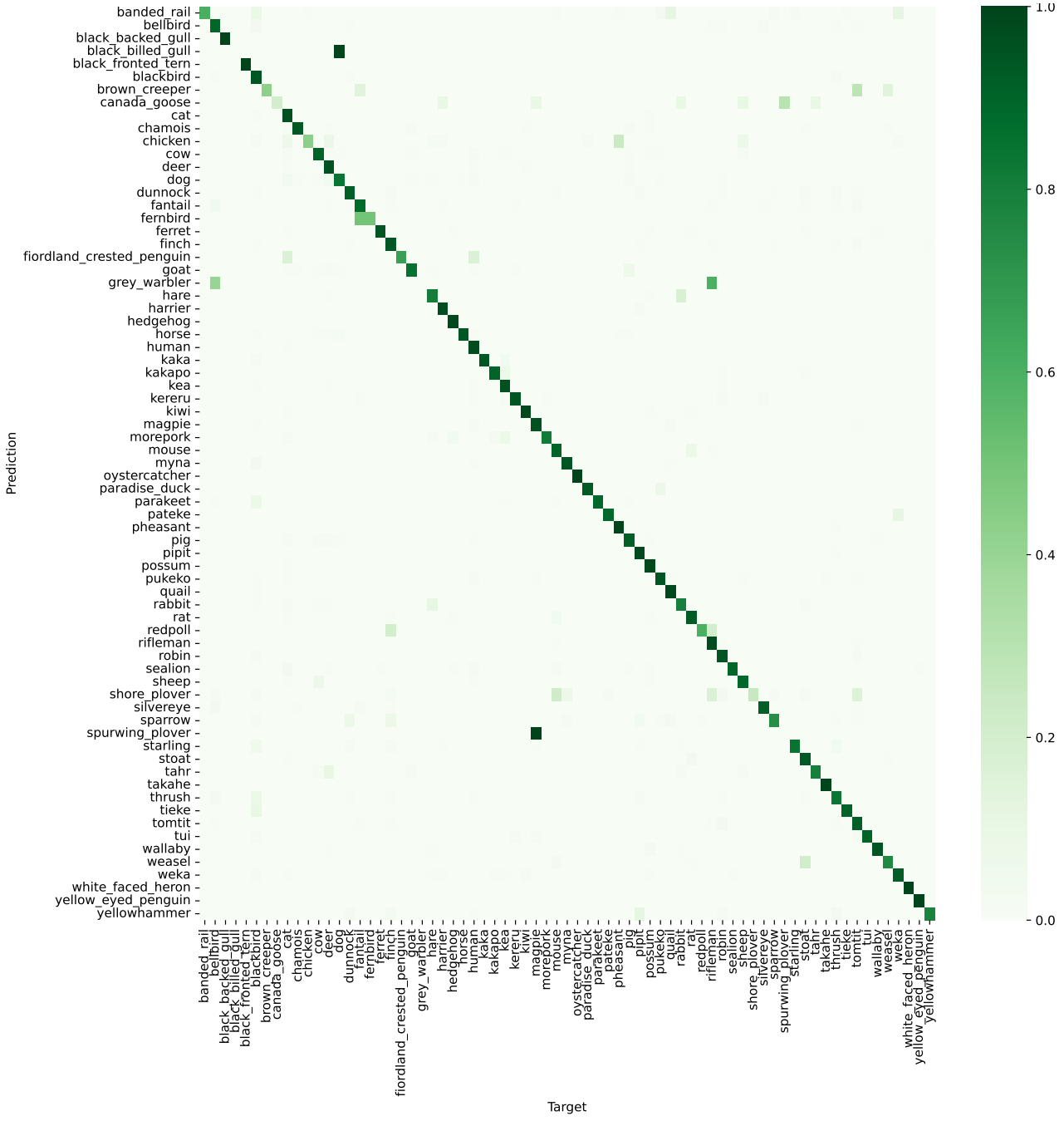


FIGURE 4: CONFUSION MATRIX FOR MAXIMUM SCORES FROM THE VALIDATION IMAGES

.

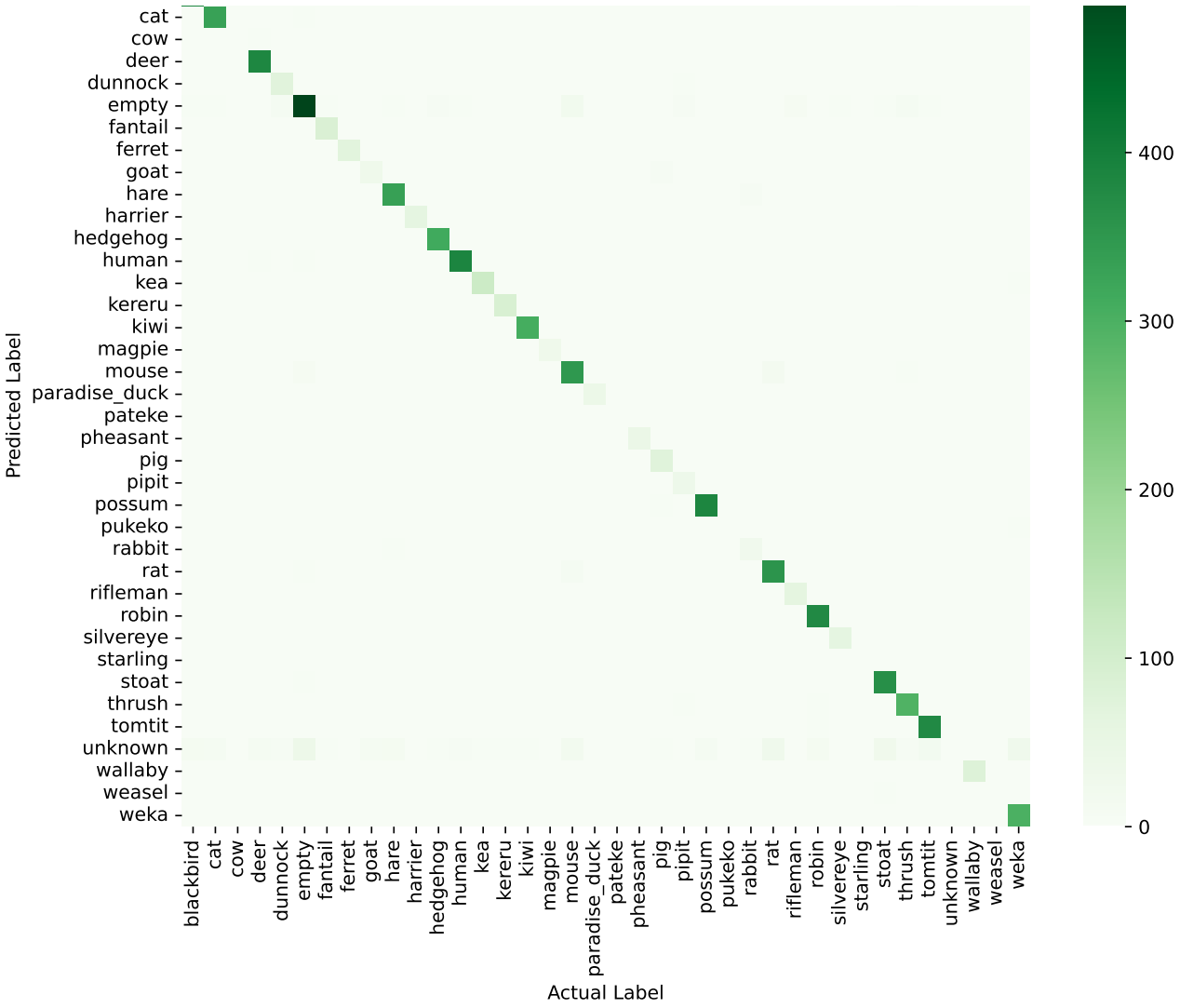


FIGURE 5: CONFUSION MATRIX USING THE 'ENCOUNTER' PREDICTION

4 RECOMMENDATIONS

This third phase of Alita's development ran into an unexpected problem quite early. The performance jumped by so much with the architectural changes that the independent test was effectively too small to convincingly investigate subsequent gains.

Whilst this is a good problem to have, further work assembling a more sensitive test set will be needed before we can reasonably continue with development.

4.1 TEST IMAGE CLASS SIZE

The current collection of test images came from 13 study areas. But once de-duplication was performed, and over-represented classes reduced to a maximum of 500 images, we were left with only 6,500 images, with only 29 classes having 30 or more images. With the model having such high accuracy, many classes in the test split produced less than 5 mistakes. Four classes actually had a perfect score. This highlights the problem we face. In order to measure incremental improvement, a much larger dataset, representing more classes will be needed.

4.2 PERFORMANCE METRICS

When we consider the goal to monitor population differences over different years, seasons and locations, the importance of consistency rather than outright performance becomes apparent.

To-date the focus has been on using ranking based metrics to assess models' abilities to predict the right classes ahead of the wrong classes. Variability in scores between different camera sites has only been measured indirectly via overall score improvement.

Future work ought to introduce a new metric to monitor and aim to improve variability for each class between camera sites within the test dataset. We also need to monitor the True Positive and True Negative score separation relative to spread in some appropriate way. We could consider variants of Cohen's D score or Fisher's discriminant ratio for this.

4.3 WEAKER CLASSES

We can still observe a significant drop-off in performance for less represented classes in figure 3. Focussing on the lowest scores, we see goat (2,082 training images), pipit (2,789 images), rifleman (6,934 images) and dunnock (4,564 images). The top 20 classes all had over 5,000 images. Further work bolstering these under-represented classes can be expected to improve their scores.

4.4 NEW RAT CLASSES

Recent fieldwork on Rakiura strongly suggests that Norway rats score quite differently to the ship rats. Currently, the vast majority of the images we have are ship rats. Whilst this is problematic for now, it suggests that if we went to the effort of developing a separate class for Norway rats the prospect of success is good. We have no similar indications for Polynesian rats, but it would make sense to investigate this at the same time if images could be sourced.

4.5 DATA SUB-SAMPLING METHOD DEVELOPMENT

The data sub-sampling methods discussed in section 2.6 were invented for this project and potentially solve a problem of interest for the wider camera trap computer-vision field.

I have made the Python code freely available on GitHub, but would like to look further into the effectiveness of these methods so that the conclusions can be published more formally. It would also be good to polish the Python methods so that they could be turned into a stand-alone library. The code is currently set up for the data structure and table formats developed for Alita. It will take some time to devise and implement more general methods.

4.6 ONGOING IMPROVEMENT TO STATE OF THE ART

As Alita Version 3.03 was under final preparation a new version of MegaDetector was released, *MDV1000 Redwood*. In addition to this, the PyTorch Wildlife repository is under constant development with multiple new models released so far.

With the combined logic in Alita, the detection step is only a small source of error, but nonetheless this remains a future prospect for improvement in both accuracy and speed.

A similar argument applies to the classification step. The EfficientNet family of networks is just one of many that could have been used as a backbone, and new networks are released regularly. Future gains in accuracy by adopting newer networks, or vision transformers are likely once we have a test set sensitive enough to measure them.

A GRAPHICAL USER INTERFACE

FIGURE 6: A SCREENSHOT OF THE GRAPHICAL USER INTERFACE

B CLASS SIZES FOR THE FULL IMAGE DATASET

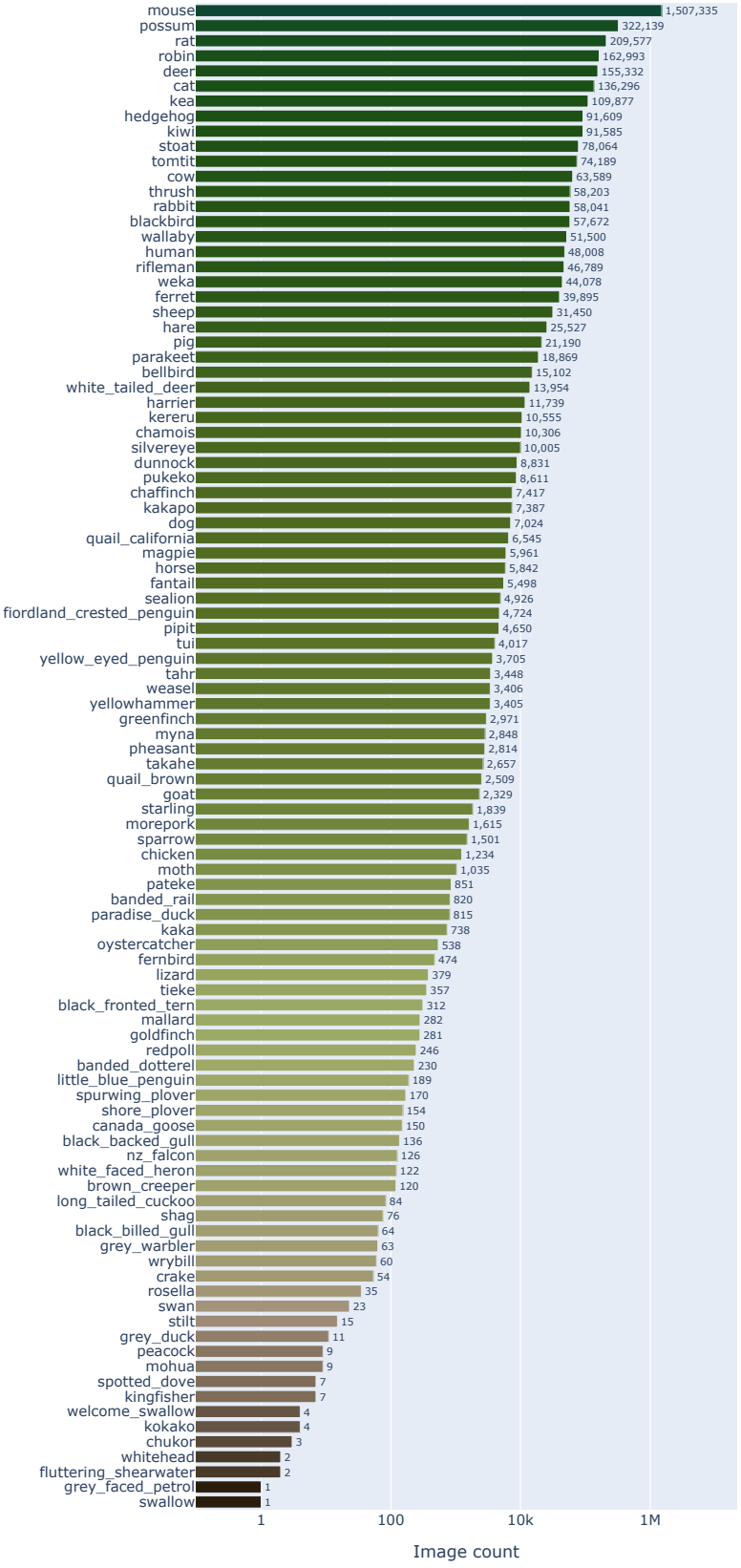


FIGURE 7: CLASS SIZE FROM THE ORIGINAL IMAGE DATASET

C CLASS SIZES FOR THE BALANCED TRAINING DATA

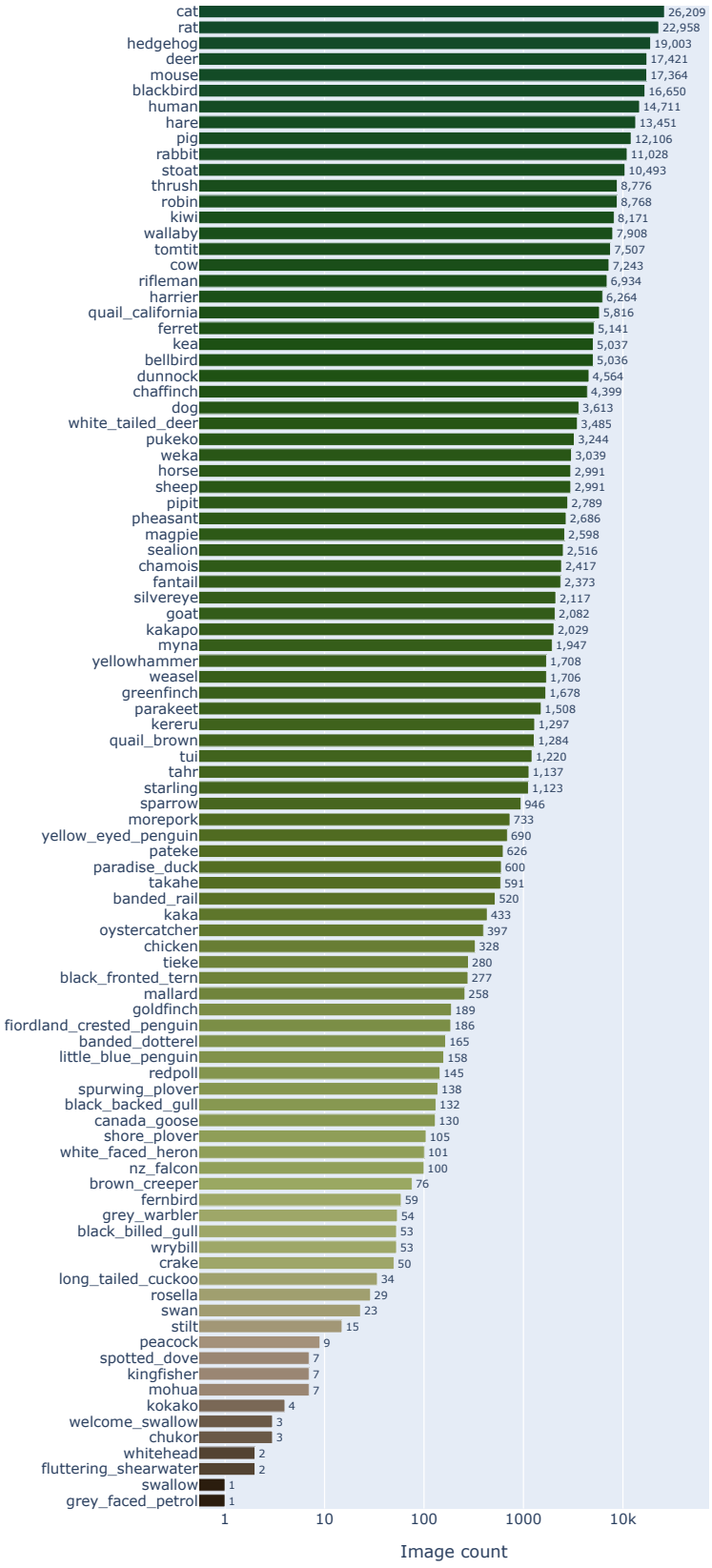


FIGURE 8: CLASS SIZES FOR THE SELECTED TRAINING IMAGE CROPS

D AVERAGE PRECISION FOR VALIDATION IMAGES

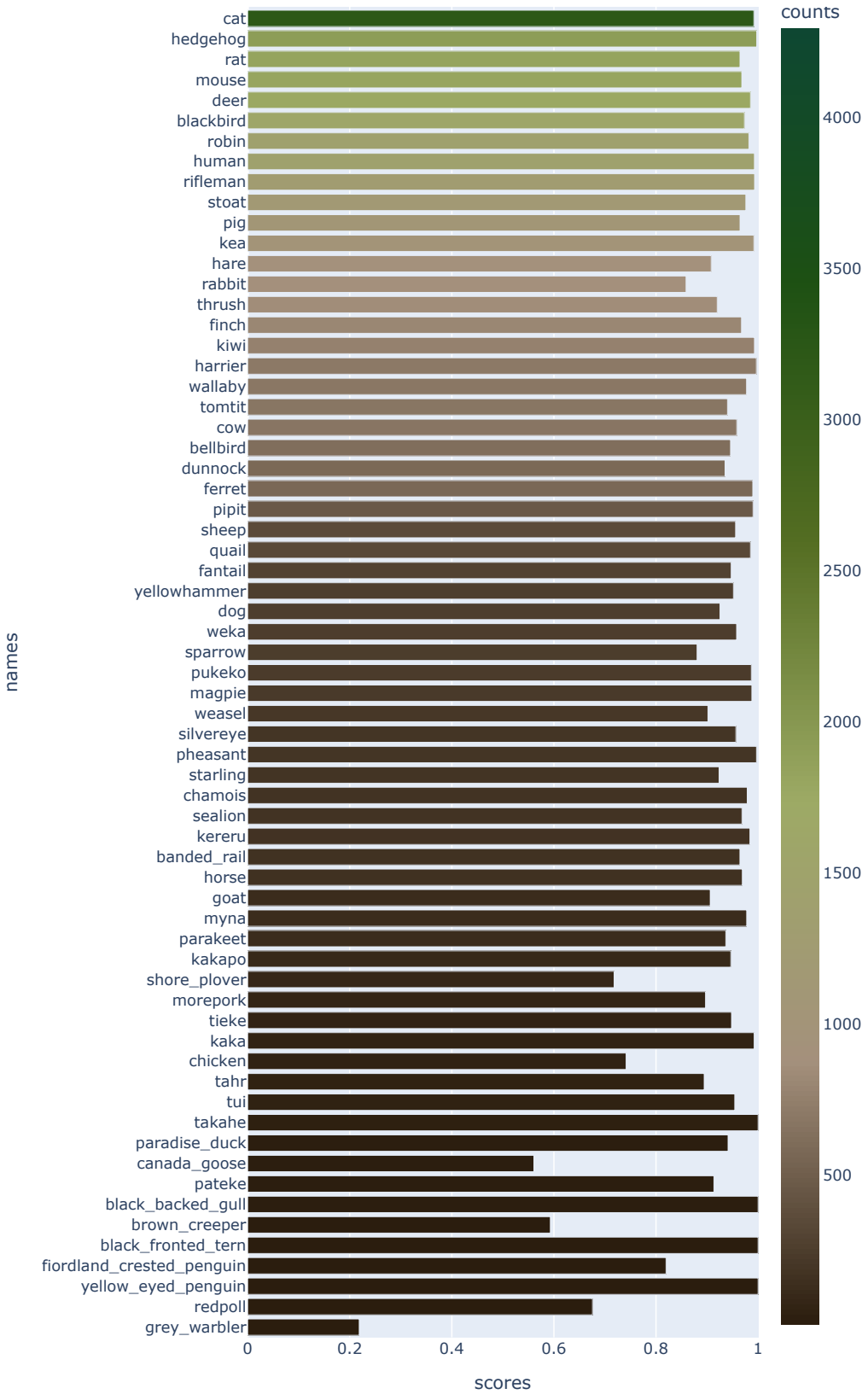


FIGURE 9: AVERAGE PRECISION SCORES FOR ALL CLASSES IN THE VALIDATION SPLIT

E PRECISION AND RECALL FOR TEST IMAGES

These were all the species that were present in the out-of-domain test set, with at least 30 images. An upper limit of 400 images per class was set and some duplicate removal performed.

TABLE 5: PRECISION AND RECALL SCORES FOR THE TEST IMAGES

Species	Precision	Recall	Target Images
blackbird	0.97	0.96	400
cat	0.99	0.96	347
deer	0.99	0.96	400
dunnock	0.93	0.81	88
empty	0.85	0.89	561
fantail	1.00,	0.95	92
ferret	0.99	1.00	70
goat	0.82	0.76	41
hare	0.98	0.95	351
harrier	1.00	1.00	62
hedgehog	1.00	0.97	323
human	0.98	0.97	400
kea	0.96	0.98	117
kereru	1.00	0.97	93
kiwi	0.99	0.99	310
mouse	0.94	0.87	400
pheasant	1.00	1.00	44
pig	0.99	0.83	88
pipit	0.97	0.74	47
possum	0.99	0.96	400
rat	0.96	0.89	400
robin	1.00	0.94	400
silvereye	1.00	0.92	66
starling	0.00	1.00	0
stoat	0.99	0.92	400
thrush	0.98	0.92	321
tomtit	0.99	0.94	400
wallaby	1.00	1.00	80
weka	1.00	0.88	340

Overall the balanced accuracy score on this data was 0.92. Note that the accuracy on empties is somewhat under-stated, as most of the false positives would have been flagged as ’unknown’, which is more useful than a false empty.

REFERENCES

[1] Microsoft, "Pytorch wildlife," https://https://microsoft.github.io/CameraTraps/, 2025, gitHub repository.

[2] G. Cocher, "Yolov5," https://github.com/ultralytics/yolov5, 2025, gitHub repository.

[3] D. Morris, "Megadetector," https://github.com/agentmorris/MegaDetector, 2025, gitHub repository.

[4] T. Gadot, Ștefan Istrate, H. Kim, D. Morris, S. Beery, T. Birch, and J. Ahumada, "To crop or not to crop: Comparing whole-image and cropped classification on a large dataset of camera trap images," *IET Computer Vision*, vol. 18, no. 8, pp. 1193–1208, 2024.

[5] O. Powell, "Predator classifier report," The Department of Conservation, Tech. Rep. R DOC 02 Rev00, December 2022.

[6] ——, "Camera trap classifier for predator detection," The Department of Conservation, Tech. Rep. DOC-7527288, January 2024.

ACKNOWLEDGEMENTS

As always, thanks to Joris Tinnemans for his ongoing hard labour collating the image datasets, and our two volunteers Jan Hewton and Jane Stevens for their tireless enthusiasm checking and labelling images.

Thanks also to the rest of the Threats Science and NPCP team for their ongoing support of this project.

Outside of DOC thanks to Dan Morris for his ongoing work on MegaDetector and LILA BC, and Saul Greenberg and his team at the University of Calgary for their work on Timelapse. Thanks also to Peter Van Lunteren for his collaboration in getting Alita integrated into his AddaxAI platform.