Automated Detection of Provisioning Events and Prey in Nestbox Video Footage

Moth

Author: Jesse W. Visser
Supervisors: Dr. Jacob W. Kamminga, Dr. Emily R. Burdfield-Steel
Critical Observer: Dr. Andreas Kamilaris
Affiliation: University of Twente, Faculty EEMCS
Location: Enschede, The Netherlands
Email: j.w.visser-3@student.utwente.nl

I. Introduction

Human actions drive a decline in biodiversity. In response, many societal organisations around the world have set conservation and preservation targets [1]. However, these targets cannot be achieved without knowledge of the current state of the ecosystem. For this purpose, various wildlife monitoring projects are started around the world to monitor biodiversity.

One such project includes new and existing 'smart' nestboxes for small cavity breeding bird species. These nestboxes are enhanced with cameras to monitor the birds and their nestlings. Often these cameras are set up to record for extended periods of time, collecting enormous amounts of data. This data is precious to researchers, as it contains valuable information about the behaviour of the monitored birds, their prey, and their nestlings. Most interesting to researchers are the moments when a parent enters the nest to feed their young, so-called provisioning events. These moments contain information about the feeding behaviour of the birds and the prey they bring into their nest.

From interviews with researchers in the field, the current workflow for analysing nestbox video footage was experienced as tedious by the researchers. Currently the processing of this data is a manual and labour-intensive task. They must manually sift through every minute of recorded video to find clips that contain interesting information. After finding these parts of the footage they will have to analyse the footage to find out what prey the parent brings into the nest. This process however is also prone to error, as it is often hard to see the difference between species of prey when footage can be short and/or blurry. Overall, the process is tedious and prone to human error.

Therefore, this paper sets out to propose a solution to significantly reduce the effort of analysing nestbox video footage. To achieve this the following research questions, need to be answered:

1. What is the fastest and most accurate method to find provisioning events in nestbox video footage?
2. What is the fastest and most accurate method to detect and classify prey in nestbox video footage?

To answer these research questions the solution proposed in this paper applies two algorithms. The first algorithm detects the provisioning events. While the second model detects and classifies the prey visible in the footage of the provisioning events. The algorithms return whether a provisioning event is happening in the frames of the nestbox video footage, and the general category of the prey provisioned by the parent.

This paper starts with an explanation of concepts that are key to understanding the algorithms used in the related work and proposed solution. Then, related work is surveyed to find out what work other researchers have done on this topic. The Dataset used in this research is introduced and the annotations are explained. This is followed by the proposed solution and experimental setup. Finally, the results are discussed, and a conclusion is made.

II. Concepts and Techniques

A. Convolutional Neural Networks

With the introduction of Convolutional Neural Networks (CNN) the accuracy of classification of image data has quickly increased compared to previous methods. First introduced by Yann Lecun and Yoshua Bengio in 1998, CNN are a type of Artificial Neural Network (ANN) [2]. These networks are especially good at processing grid-like data, such as time-series and images. A CNN consists of layers of 'neurons', which each have learnable parameters and in sequence execute computations on the input data. A CNN often include several basic layers but are not limited to these basic building blocks. A CNN classically contains the following layers:

- An input layer consists of a matrix of neurons representing the raw image data. This matrix has the dimensions W×H×C, where in relation to the image, W is width, H is height and C is colour. Colour will consist of one or more channels for each colour value of the pixels in the image. Most common are RGB images, which have three colour channels.
- A convolution layer consists of neurons that each take the dot product of a restricted section of values from the preceding layer (the receptive field) and a matrix of learnable parameters (the kernel). By arranging the layers in a certain way, the network can first detect simple features, such as shapes, and then more high-level features, such as object, animals, or plants.
- A non-linearity layer, also known as an activation function, helps introduce non-linearity to the linear convolutional operations, which is needed to create non-linear decision boundaries. Without this layer a deep CNN network would perform as if it were an equivalent single convolutional layer.
- A pooling layer derives a summary statistic of the output of preceding layers and can be placed in various locations in a network to help reduce the spatial size of the representation. This decreases the number of weights and the subsequent computation needed for a network.

- A flatten layer can fit the multi-dimensional output from the preceding layers into a one-dimensional vector which can be used as the input for a fully connected layer.
- A fully connected layer is often the last layer of a network. The layer is fully connected, which means that every neuron on this layer is connected to all the neurons in the preceding layer. This layer maps the input to the required output, such as a classification.

B. Hyperparameters

In the training process of a Neural Network there are various hyperparameters which can be tuned to reach the optimal performance of a model as quickly as possible. Using an optimizer algorithm during training, local minima in the weight values can be reached faster. Algorithms such as Stochastic Gradient Decent (SGD) or Adaptive Moment Estimation (Adam) are common algorithms to optimize CNN training. It has also become common to use momentum to reach local minima in the optimization algorithm. As recommended by Sutskever in [3], specifically using the Nesterov algorithm [4] can significantly speed up the learning process. Another hyperparameter is learning rate, which can be either constant or varied. Which means that there is either a set learning rate to use during the entire training process or changing the learning rate during the training process.

C. Transfer Learning

Transfer learning [5] is a technique which has been used in recent years to increase the training speed and performance of CNN models. Instead of initialising the model with random weight, the model is 'pre-training' on a large dataset, such as ImageNet [6]. This way the model will learn generalisable feature representations. By 're-training' the network on a more specific dataset the training process will require less data and time. On small datasets, this technique also helps against overfitting. A model that is overfitting is extremely accurate on the trained data but does not perform well on new testing data. The process of transfer learning does require the training data of the new dataset to be similar in dimensions to the dataset the model was pre-trained on.

D. Tensorflow

With the release of TensorFlow [7] around 2015, the implementation of CNN and other Machine Learning (ML) techniques became more accessible. TensorFlow is an open-source project built at Google. It features an interface to express ML algorithms and an implementation to execute these algorithms in various programming environments. For this reason, it has often been used by researchers and in various application to implement CNN architectures. Popular architectures are implemented in the TensorFlow API and come with the install of the package. For fast comparisons of different architectures, the TensorFlow Python package is used for the implementation and training of the CNN models tested in this research paper.

E. Architectures

Various CNN architectures have been designed for the task of classifying image data. AlexNet [6] is often cited as one of the first network architectures to reach human classification accuracy when it won the ImageNet classification challenge in 2012. Subsequent architectures kept adding layers, but ultimately suffered from the exploding/vanishing gradient problem, where the weight gradient quickly approaches zero or explodes into extremely large values. However, in 2016, the authors of [8] introduced ResNet, a deep neural network which no longer suffered from the exploding/vanishing gradient problem by implementing 'residual blocks'. More recent model architectures such as MobileNet [9]–[11], YOLO [12]–[15] and EfficientNet [16], [17] combine various techniques to further increase accuracy while reducing model size and even achieving real time performance on smartphones and other edge devices.

III. Related Work

Classically, nestbox video footage is annotated manually. Which can be read from the methods sections of many papers published on the analysis of these types of footage. This was further confirmed by interviews with authors of some of these papers.

In [18] E. Pagani-Núñez and J.C. Senar monitored 182 nestboxes inhabited by Great Tits (*Parus Major)* near Barcelona, Spain. They installed camouflaged Micro-D cameras inside the nestboxes when nestlings were 10-16 days old. Starting recording one day after installing and comprising footage between 07:00 and 14:00. The amount of recorded data was limited using a motion sensor to detect movement at the entrance of the nestbox. The footage was then manually analysed, and the prey type, size and time of provisioning was recorded. They identified three categories of prey: *caterpillars, spiders,* and *others*. The prey size was categorised according to a semi-quantitative scale: *small, medium, or large*. From interviews with one of the authors, J.C. Senar, it became clear that the analysis process took several months. Where they had to tediously go through the large amount of data collected during the monitoring period.

From interviews with doctoral researcher at the University of Helsinki T.M. Abaurrea it became clear that they too faced similar problems with the analysis of their nestbox video datasets. Having to manually go through hours of continuous recorded footage to find provisioning events and annotate information relevant to their research.

Fig. 1 Example frame from the dataset labeled as provisioining.

Fig. 2 Example frame from the dataset labeled as not-provisioining

More research is done on automated analysis of camera traps for mammals and other larger wildlife. In [19] M. Willi R. Pitman, R. Cardoso et al. implemented two model. First, they managed to identify whether an animal was spotted in an image from savanna and forest snapshots with 91.2% accuracy. Then they were able to classify specific species from a total of 17 species with accuracies between 88.7% and 92.7% depending on the species.

Similarly, in [19] S. Schneider, G. Taylor and S. Kremer created the Gold Standard Snapshot Serengeti dataset, based on the Snapshot Serengeti dataset [20], which contains forty mammalian species of the African savanna. On this dataset they were able to achieve a 76.7% (±8.3) detection accuracy with a Faster R-CNN [21] model. Using a detection model instead of simple classification allows for the identification of multiple animals in the same image but required more detailed annotations.

In [22] S. Kennelly and R. Green try to solve a problem similar to the detection of provisioning events in nestbox video footage. They implemented a CNN model to detect whenever a bird is sitting at a bird feeder. They collected a dataset of 5,375 images from two different bird feeders. The data was labelled either: *Parakeet, Other Bird, Empty* or *Other*. Because the dataset is small the authors used data augmentation and transfer learning to avoid overfitting the model. Using a two-pipeline approach with the ResNet18 and VGG16 [23] architecture, the authors reached a 99.12% true positive accuracy on the Parakeet class.

Most relevant is the paper by H. Williams, L. Matott and R. DeLeon [24], where they detect whenever a Purple Martin (*Progne subis*) provisions its nest. They collected a dataset of 83.254 images taken from a total of 13,000 hours of full colour footage from twenty different nestboxes. The original images were 1920×1080 pixels but were reduced to 0.1 times the size to speed up the training process. The images were annotated either: *Zero Birds*, *One Bird,* or *Two Birds*. Although, they found it rare for two birds to provision at the same time and even more rare for a parent to enter the nest and not provision its young. With this data the authors trained a CNN model based on the VGG16 architecture and reached an accuracy between 77% and 88% for the analysis of the provisioning data. Although technically performing worse than a human could, the accuracy is sufficient to determine the feeding frequency according to their Monte Carlo simulation [25].

In researching related work, it became apparent that at this point little research has been done regarding object detection and classification in relation to the analysis of smart nestbox video footage. Despite this, knowledge gained in research in related fields is transferable to this application.

IV. Dataset

To the best of our knowledge, there is no benchmark dataset containing images of nestbox cameras. The dataset used in this paper was shared by doctoral researcher T.M. Abaurrea. The dataset contains 48 hours of 2304×1296 pixels, greyscale nestbox camera video of Redstarts (*Phoenicurus phoenicurus*) and their young. At roughly 30 frames per second (fps), this is a total of 5,230,110 images. It is important to note that the dataset is limited to only one bird species and that the footage is taken with a specific angle relative to the inside of the nestbox. This might limit the generalisability of the final trained model. An accompanying spreadsheet contains annotations of the footage with the entry time of the parent bird and the duration of the provisioning event. Additional labelling files contain a bounding box and category of the prey the parent brings into the nest when provisioning their young. The following sections explain how the data was processed to be used in the training process of the proposed models.

A. Preprocessing

The models include pre-processing steps that resize the images to the appropriate size specified by the model architecture and pre-training dataset used for transfer learning. This is 224×224 pixels for the classification models and 640×640 pixels for the detection model.

B. Provisioning event classification dataset

The MoviePy [26] library was used to load the source videos in a Python script, where the video was split into individual

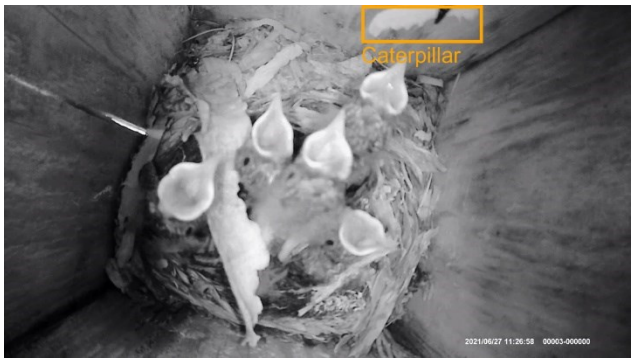

Fig. 3 Video frame with bounding box and label Caterpillar for prey.

Fig. 4 Video frame with bounding box and label Moth for prey.

frames. This step is needed to be able to create a static dataset of images to train the CNN models on. The script loops over the number of frames in a video and exports the frames to specific directories based on the provisioning event time annotations.

The frames within the range of a provisioning event time annotation are exported to a directory named *provisioning* and all other frames are exported to a directory named *non-provisioning*. An example of what frames are annotated as either *provisioning* or *not-provisioning* can be seen in **Fig. 1** and **Fig. 2**, respectively.

To compensate for inaccuracy in the annotations the five seconds of footage before and the 5 second after each provisioning event are discarded. This is a total of 300 discarded frames per provisioning event, which could have otherwise been ambiguous to which class they should belong. Finally, the annotations are manually checked for erroneous categorization. The raw dataset is imbalanced as the provisioning events take up only a small section of the total videos. Therefore, the number of images in each class is equalized by randomly sampling an equal number of images from each class.

The final dataset for the binary classification of provisioning event frames consists of 80,000 images evenly split over the *provisioning* and *not-provisioning* classes. The images are split into 80% training, 15% validation and 5% test sets. The final division of the provisioning event dataset can be seen in **Table I**.

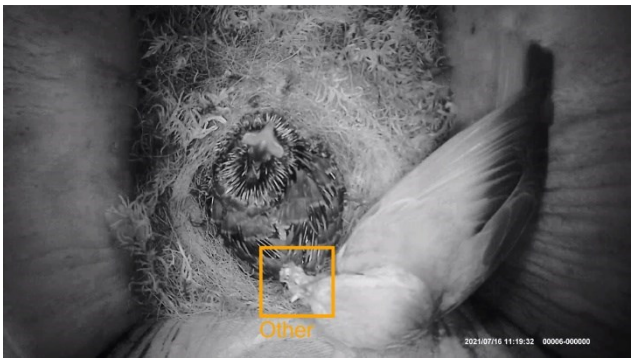

Fig. 5 Video frame with bounding box and label Other for prey.

Fig. 6 Video frame with bounding box and label Spider for prey.

TABLE I PROVISIONING EVENT CLASSIFICATION DATASET SPLIT

Class	Total	Training	Validation	Test
Provisioning	40,000	32,000	6,000	2,000
Not-provisioning	40,000	32,000	6,000	2,000

C. Prey detection dataset

For the detection and classification of prey images from the *provisioning* class of the provisioning event dataset are annotated further. VOTT [27] is an image and video annotation tool developed by Microsoft. It was used to annotate a subset of 1270 frames from the nestbox video dataset with a bounding box for the location of the prey and a label for the class of the prey. The label classes consist of the three most frequently provisioned prey: *Caterpillar, Moth,* and *Spider*. All other prey type annotations are grouped into the class *Other*. The annotations are split into 80% training, 15% validation and 5% test sets. The final division of the prey annotations can be seen in **Table II**. Examples of the annotated images can be seen in **Fig. 3**. The VOTT label export files are then converted to the YOLO annotation format for compatibility with the detection model training setup. The YOLO annotation format consists of text files accompanying the image dataset. Each row in the annotation files contains the coordinates of a bounding box of an object in the image.

TABLE II PREY DETECTION DATSET SPLIT

Class	Total	Training	Validation	Test
Caterpillar	332	258	58	15
Moth	203	157	33	13
Spider	489	390	69	30
Other	246	208	31	7

V. METHODS

The proposed solution combines two models to analyse the nestbox video footage. The first model classifies the provisioning event frames while the second model detects and classifies the prey in the provisioning event frames.

A. Classification of Provisioning Events

To detect the provisioning events in the dataset, a CNN model was trained to classify the individual frames of the footage as either *provisioning* or *not-provisioning*, representing whenever the parent bird is in the nestbox or not. It was found that during the period the video was taken it is unlikely for the parent bird to enter the nest without provisioning its young. It was also found to be unlikely for there to be more than one parent in the nest at the same time.

With these assumptions a provisioning events can be determined simply by the presents of a parent bird. Therefore, a binary classification model suffices for detecting provisioning events in this dataset.

To answer the first research question and find the fastest and most accurate CNN architecture for the provisioning event classification model, five CNN architectures are compared. Different generations of the EfficientNet and MobileNet architecture are trained on the provisioning event dataset to compare their accuracy and inference speed. From the EfficientNet architectures the first and second version were tested. Choosing the smallest version labelled B0 to be more similar in computational size to the different generations of MobileNet. From the MobileNet architectures the first, second and third generation were tested. Using the smallest version of the third-generation architecture, which is also more similar in computational size to the other tested models.

The training and evaluation setup for the classification models is implemented using the TensorFlow API in Python using GPU acceleration. The model architectures are loaded from the TensorFlow Keras API [28]. The models include a pre-processing layer to resize images to 224×224 pixels and are initialized with weights pre-trained on the ImageNet dataset [6]. The models were then trained on the provisioning event frames.

All the binary classification models have been trained for 100 epochs or until validation metrics were no longer improving. Each training epoch consisting of 500 steps with batches of 128 images per step. The training procedure used the NAdam optimizer, which incorporates Nesterov momentum into the Adam optimizer. With the following hyperparameter values: learning rate=0.001, $\beta1$=0.9, $\beta2$=0.999 and $\varepsilon=1e^{-6}$.

B. Detection and Classification of Prey

Introduced in 2021 by Chien-Yao Wang et al., You Only Learn One Representation (YOLOR) is a cutting-edge real-time detection model architecture. This model architecture was chosen as it is shown to have significantly faster interference compared to other state of the art architectures, while reaching similarly competitive accuracy on benchmark datasets [29].

To answer the second research question a model with the YOLOR-P6 architecture is trained to predict the location and class of the prey in provisioning event frames. The model is trained and evaluated using the implementation by the authors themselves [30]. The model includes a pre-processing step to resize images to 640×640 pixels. The model is initialized with weights pretrained on the Common Objects in Context (COCO) dataset [31], consisting of over 200,000 images and 1.5 million object instances categorized into 80 object categories (such as person, bicycle, bird, etc.). The model is then trained on the prey annotations using the finetuned hyperparameters from the authors themselves which can be found included in the training code repository.

C. Evaluation Metrics

There are various metrics that can be used to evaluate the performance of a trained model. The metrics used between classification models and detection models differ slightly. The metrics used in this research paper are explained below.

1) General model metrics

- Precision is the proportion of all examples above a given rank which are from the positive class and can be written as $\text{Precision} = \frac{\text{TP}}{(\text{TP}+\text{FP})}$. Where TP is True Positive, and FP is False Positive.
- Recall is defined as the proportion of all positive examples ranked above a given rank and can be written as $\text{Recall} = \frac{\text{TP}}{(\text{TP}+\text{FN})}$. Where TP is True Positive, and FN is False Negative.
- Inference speed measures how quickly a model can classify a given image. This is often measured in frames per second (fps). This measure depends on the implementation of the model and the hardware it is running on.

2) Classification model metrics

- Accuracy is calculated as the ratio between the number of correct predictions and the total number of predictions and can be written as $\text{Accuracy} = \frac{\text{TP} + \text{TN}}{(\text{TP}+\text{TN}+\text{FP}+\text{FN})}$ .Where TP is True Positive, TN is True Negative, FP is False Positive, and FN is False Negative.

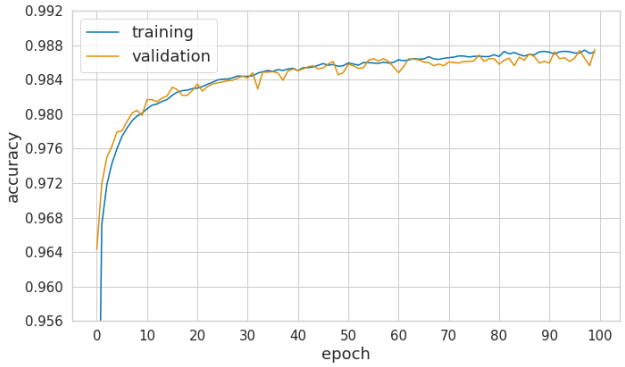

Fig. 7 Training and validation accuracy of MobileNetV3Small classification model on the provisioning event annotations.

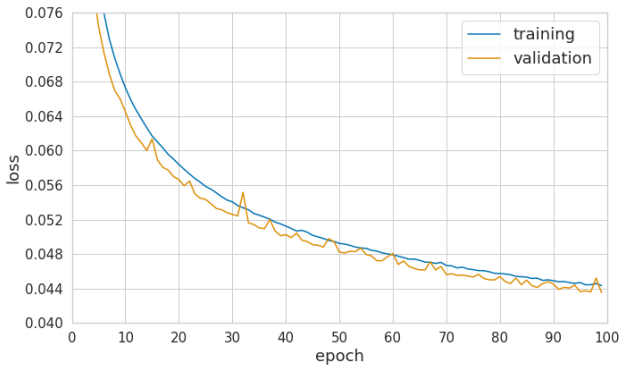

Fig. 8 Training and validation loss of MobileNetV3Small classification model on the provisioning event annotations.

- Receiver Operator Characteristic curve (ROC) is a graph of the performance of a model at all confidence score thresholds. Plotting the True Positive Rate (TPR) against the False Positive Rate (FPR) at different thresholds.
- Area Under Curve (AUC) of the ROC is the entire two-dimensional area underneath the ROC curve. This metric provides an aggregate measure of performance across every classification threshold.

3) Detection model metrics

- Intersection over Union (IoU) gives the accuracy of the prediction of bounding boxes by the ratio between the intersection and the union of the predicted boxes and the annotated boxes.
- Precision/Recall curve is a graph that shows the trade-off between precision and recall as a function of the model confidence score threshold.
- Mean Average Precision (mAP) was formalized in the 2007 PASCAL Visual Objects Classes (VOC) challenge [32]. First the Average Precision (AP) is calculated, taking the AUC of the precision/recall curve using an IoU value of 0.5 as the confidence score threshold. The mAP is then calculated as the average AP over all classes. Notably, the MS COCO challenge [31] uses the average mAP over 10 different IoU thresholds from 0.5 to 0.95.

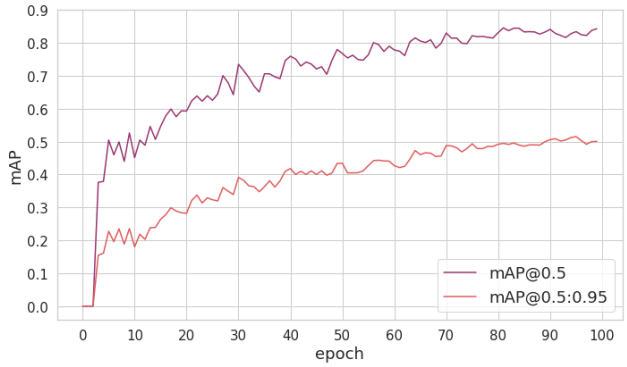

Fig. 9 The mAP at IoU@0.5 and IoU@0.5:0.95 during training of YOLOR detection model on the prey annotations.

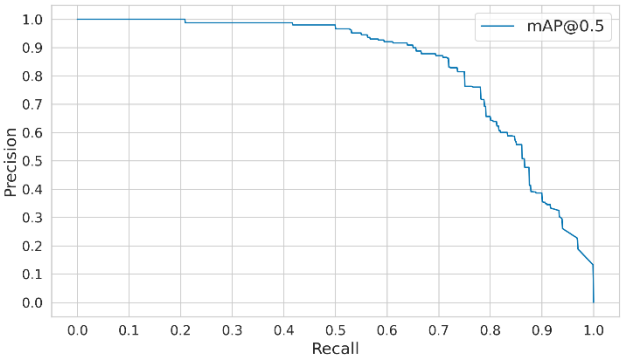

Fig. 10 The precision/recall curve of YOLOR trained on the prey annotations.

VI. Results

A. Experiment Hardware

The experiments are caried out in Windows Subsystem for Linux (WSL) using CUDA 11.7 drivers on a Windows system featuring an Nvidia RTX2080 maxQ GPU, an Intel core i7-9750H CPU at 2.60GHz and 32 GB of random-access memory. The models are implemented and run using Python version 3.8 in Anaconda3 for Linux.

B. Detection of Provisioning Events

Table III summarizes the results of the experiment. The first results column shows the accuracy on the evaluation split of the dataset. The second column shows the AUC of the ROC curve of the model predictions on the evaluation split of the dataset. The last column shows the inference speed of the models per image with batches of 128 on the evaluation split of the dataset using the experiment hardware.

Table III Provisioning Event Model Metrics

Model	Accuracy	AUC	Inference
EfficientNetV2-B0	99.04%	99.51%	135fps
MobileNetV3Small	**98.75%**	**99.29%**	**290fps**
EfficientNetB0	98.73%	99.49%	94fps
MobileNet	96.81%	98.02%	181fps
MobileNetV2	95.77%	95.22%	151fps

Although EfficientNetV2-B0 technically reached the highest accuracy, MobileNetV3Small reached significantly faster inference speed while getting extremely close to the

Fig. 11 Example of double detection fault.

accuracy of EfficientNetV2-B0. **Fig. 7** and **Fig. 8** show that the performance of the model on the evaluation split of the dataset is just as good as the performance on the training split of the dataset. This means that the model is not overfitting or underfitting.

C. Detection and Classification of Prey

In the experiment YOLOR initialised with weights pre-trained on the COCO dataset was re-trained on the prey annotations. The training mAP at an IoU threshold equal to 0.5 and between 0.5 and 0.95 is shown in **Fig. 9**. The final model reached a mAP of 83.6% at an IoU threshold equal to 0.5 and 51.4% at IoU between 0.5 and 0.95. This can also be seen in the precision/recall graph in **Fig. 10**. **Table IV** shows the precision, recall and mAP for each class individually. The model can run detections at 90fps on the experiment hardware.

TABLE IV PREY MODEL METRICS

Class	Precision	Recall	mAP@0.5	mAP@0.5:0.95
All	0.529	0.861	0.836	0.514
Caterpillar	0.576	0.833	0.826	0.506
Moth	0.674	0.939	0.947	0.649
Spider	0.553	0.824	0.800	0.399
Other	0.314	0.846	0.770	0.503

VII. DISCUSSION

By the AUC value of the ROC of the classification models, it is shown that the models have both very hight precision and recall. Meaning the models will correctly classify most samples while finding little to no false positives or false negatives. In the application of analysing nestbox video footage it is most important that all occurrences are detected correctly. It is easier to remove false positives than to find missing occurrences.

Similarly, **Table IV** shows that the prey detection model has quite high recall, but low precision. Which means it will correctly classify most samples but will give some false positives. Which in the case of detection might also mean a correct classification but at an incorrect location relative to the IoU threshold. Which can be seen in the mAP@0.5:0.95, where the average it taken from 10 thresholds of IoU. In case the model would be used to find the exact location of the prey in the frame it would not be precise enough to correctly segment the prey.

Table IV also shows that the *Other* class has a significantly lower mAP than the other classes, which may be explained by the greater variance compared to the other classes. The *Spider* class also has low accuracy at mAP@0.5:0.95. This may be explained by the low contrast with the background but requires more research. Sometimes the model also confuses two halves of a prey for multiple separate prey. This can be seen in **Fig. 11**.

The number of prey annotations is low. Overall, the detection model should improve significantly with more annotation data and more balanced classes for training.

VIII. CONCLUSION

The MobileNetV3Small architecture was trained on nestbox video footage and was found to be a fast and accurate model to classify provisioning event frames. Reaching an accuracy of 98.75% and running inferences at 290fps on the experiment hardware. This means that the trained model can quickly and accurately predict provisioning events in nestbox video footage. By integrating this model into an analysis software, the feeding events from a video of an hour at 30fps could be correctly detected in just over 6 minutes. Work which would take a researcher at least an hour, if not more.

The YOLOR model architecture trained for the detection and classification of the prey in the provisioning frames into four broad categories reaches a mAP of 83.6% at an IoU threshold equal to 0.5 and processing at 90fps on the experiment hardware. This means that when the model is integrated in an analysis software it could process provisioning event footage at 30 fps 3 times faster than real-time. Finding the general location of the prey and classifying it into one of 5 broad classes. This further reduces the work researchers would need to do to analyse nestbox video footage.

By combining the two models presented in this paper into an analysis software, the solution could quickly and accurately classify provisioning event frames and predict the broad class and general location of the prey provisioned. Greatly improving the workflow of analysis nestbox video footage for researchers.

IX. FUTURE WORK

There are various steps that could be taken to create an even better solution. Especially the dataset and annotations could be improved.

Future research should be done on the amount of data needed to train the provisioning event classification model. The number of images used to train the provisioning event classification model is relatively large and it would be useful

to know the minimum amount of data needed to reach a satisfying accuracy.

With more prey annotation it might also be possible to get more specific classification and segmentations of the prey. Finding more specific species and adding more fine-grained classes. Colour information might also help to define more fine-grained prey classes.

Lastly, the dataset is limited in its variance. The footage is only recorded for one species of birds and does not include much variation in filming angle and lighting conditions. An effort could be made to create a more varied benchmark dataset from nestbox video footage from many different nestboxes and bird species.

ACKNOWLEDGMENTS

I would like to thank: Dr. Jacob W. Kamminga and Dr. Emily R. Burdfield-Steel for actively supervising this thesis. Dr. Andreas Kamilaris for being the critical observer of this thesis. Doctoral researcher Teresa M. Abaurrea for meeting me and sharing her video dataset and annotations. Justine Loof for adding additional annotations to the dataset. Dr. Juan C. Senar for meeting me and sharing further information about their research methods.

REFERENCES

[1] S. Díaz *et al.*, "Pervasive human-driven decline of life on Earth points to the need for transformative change," *Science (1979)*, vol. 366, no. 6471, Dec. 2019, doi: 10.1126/science.aax3100.

[2] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015, doi: 10.1038/NATURE14539.

[3] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," 2013.

[4] Z. Yang, W. Bao, D. Yuan, N. H. Tran, and A. Y. Zomaya, "Federated Learning with Nesterov Accelerated Gradient Momentum Method," Sep. 2020, [Online]. Available: http://arxiv.org/abs/2009.08716

[5] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?," Nov. 2014, [Online]. Available: http://arxiv.org/abs/1411.1792

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, Kai Li, and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2009, pp. 248–255. doi: 10.1109/CVPR.2009.5206848.

[7] M. Abadi *et al.*, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems," Mar. 2016, Accessed: Apr. 26, 2022. [Online]. Available: http://arxiv.org/abs/1603.04467

[8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," 2016. [Online]. Available: http://image-net.org/challenges/LSVRC/2015/

[9] A. G. Howard *et al.*, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," Apr. 2017, [Online]. Available: http://arxiv.org/abs/1704.04861

[10] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," 2018.

[11] A. Howard *et al.*, "Searching for MobileNetV3," May 2019, [Online]. Available: http://arxiv.org/abs/1905.02244

[12] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," Jun. 2016. doi: 10.1109/CVPR.2016.91.

[13] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," 2017. [Online]. Available: http://pjreddie.com/yolo9000/

[14] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," Apr. 2018, [Online]. Available: http://arxiv.org/abs/1804.02767

[15] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," Apr. 2020, doi: 10.48550/arxiv.2004.10934.

[16] M. Tan and Q. v. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," *36th International Conference on Machine Learning, ICML 2019*, vol. 2019-June, pp. 10691–10700, May 2019, doi: 10.48550/arxiv.1905.11946.

[17] M. Tan and Q. v. Le, "EfficientNetV2: Smaller Models and Faster Training," Apr. 2021, doi: 10.48550/arxiv.2104.00298.

[18] E. Pagani-Núñez and J. C. Senar, "One Hour of Sampling is Enough: Great Tit Parus major Parents Feed Their Nestlings Consistently Across Time," *Acta Ornithologica*, vol. 48, no. 2, pp. 194–200, Jun. 2013, doi: 10.3161/000164513X678847.

[19] M. Willi *et al.*, "Identifying animal species in camera trap images using deep learning and citizen science," *Methods in Ecology and Evolution*, vol. 10, no. 1, pp. 80–91, Jan. 2019, doi: 10.1111/2041-210X.13099.

[20] S. Schneider, G. W. Taylor, and S. Kremer, "Deep Learning Object Detection Methods for Ecological Camera Trap Data," in *2018 15th Conference on Computer and Robot Vision (CRV)*, May 2018, pp. 321–328. doi: 10.1109/CRV.2018.00052.

[21] A. Swanson, M. Kosmala, C. Lintott, R. Simpson, A. Smith, and C. Packer, "Snapshot Serengeti, high-frequency annotated camera trap images of 40 mammalian species in an African savanna," *Scientific Data*, vol. 2, no. 1, p. 150026, Dec. 2015, doi: 10.1038/sdata.2015.26.

[22] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks." [Online]. Available: http://image-net.org/challenges/LSVRC/2015/results

[23] S. Kennelly and R. Green, “Classifying Bird Feeder Photos,” in *2020 35th International Conference on Image and Vision Computing New Zealand (IVCNZ)*, Nov. 2020, pp. 1–6. doi: 10.1109/IVCNZ51579.2020.9290682.

[24] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” 2015. [Online]. Available: http://www.robots.ox.ac.uk/

[25] H. M. Williams, L. S. Matott, and R. L. DeLeon, “Automated Deep Learning Analysis of Purple Martin Videos Depicting Incubation and Provisioning,” in *Proceedings of the Practice and Experience in Advanced Research Computing on Rise of the Machines (learning)*, Jul. 2019, pp. 1–7. doi: 10.1145/3332186.3332194.

[26] R. L. Harrison, “Introduction To Monte Carlo Simulation,” *AIP Conf Proc*, vol. 1204, p. 17, 2010, doi: 10.1063/1.3295638.

[27] “Zulko/moviepy: Video editing with Python.” https://github.com/Zulko/moviepy (accessed Jul. 10, 2022).

[28] “microsoft/VoTT: Visual Object Tagging Tool: An electron app for building end to end Object Detection Models from Images and Videos.” https://github.com/Microsoft/VoTT (accessed Jul. 10, 2022).

[29] “keras-team/keras: Deep Learning for humans.” https://github.com/keras-team/keras (accessed Jul. 10, 2022).

[30] C.-Y. Wang, I.-H. Yeh, and H.-Y. M. Liao, “You Only Learn One Representation: Unified Network for Multiple Tasks,” May 2021, [Online]. Available: http://arxiv.org/abs/2105.04206

[31] C.-Y. Wang, I.-H. Yeh, and H.-Y. M. Liao, “WongKinYiu/yolor: implementation of paper - You Only Learn One Representation: Unified Network for Multiple Tasks,” 2021. https://github.com/WongKinYiu/yolor (accessed Jul. 10, 2022).

[32] T.-Y. Lin *et al.*, “Microsoft COCO: Common Objects in Context,” May 2014, [Online]. Available: http://arxiv.org/abs/1405.0312

[33] M. Everingham *et al.*, “The PASCAL Visual Object Classes (VOC) Challenge”.