University of Glasgow | School of Computing Science

Hungry birds - Classification of blue tit nestling diet from video footage using machine learning

Emma Poliakova (2389098p)

April 13, 2023

ABSTRACT

Computer vision in ecology is becoming increasingly popular as an efficient tool to analyze large data sets from wild populations. For instance, automating classification of behaviours from video footage can increase the breadth, duration, and repeatability of ecological studies, and removes the image processing bottleneck. Additionally, automated systems provide a non-invasive way to observe wildlife and avoid the disruption created by catching and sampling individual birds. This work presents a novel machine-learning pipeline to quantify the food provisioning rate of adult blue tits to their nestlings and classify the diet of nestlings. The pipeline is capable of detecting a bird entering a nest box, locating eyes and beak in the image, and classifying the food item carried. In the long run, this work can uncover evidence of action needed to aid birds living in our cities. The networks used to build this pipeline are YOLOv5, Keypoint R-CNN, and MobileNetv3. The food classifier achieved 69% accuracy. The bird detection component allows the researcher to look only at the relevant images of a bird carrying a food item instead of having to search for the visits in long videos.

1. INTRODUCTION

Urbanization is a growing global phenomenon but there are few bird species that thrive in the urban environment [1]. Cities can become ecological traps for birds, seemingly providing better nesting sites, such as various cavities in buildings, but lower quality food compared to woodland sites, leading to smaller reproductive success [1]. To determine if urban habitats need optimizing for biodiversity conservation, it is necessary to understand the challenges the urban adapters face. Several ecology studies [1, 2, 3] regarding urbanization selected the blue tit as the target species because nestling fitness, i.e. the weight and size corresponding to its age, is known to be tightly linked to a highly specialised food source: caterpillars. These studies have shown that the food availability for nestlings may be constraining urban birds, leading to smaller clutches in the urban sites and also decreased fitness [3].

The breeding success of blue tits is being monitored across the Glasgow urban gradient [4]. The gradient starts in the center of Glasgow which represents the most urbanized site and continues up to Loch Lomond, which is the woodland site. Almost 500 nest boxes have been placed during this ongoing research. Video-recording systems shown in figure 1 were installed in blue tit nest boxes located in Kelvingrove Park in Glasgow and the Scottish Centre for Ecology and the Natural Environment to observe the parental provisioning behaviour and the diet provided to nestlings. Until now, the ecologists studying the diet of blue tit nestlings would manually record the visits and food items in a Microsoft Excel spreadsheet. The type of prey was classified as caterpillars, insects, other, or unidentified. This created tedious labour as the videos, see figure 2, would take many hours to work through. Therefore, only certain times of each day have been annotated, specifically between $7:00-7:30$ a.m. and $8:00-8:30$ p.m. These times were selected due to the highest bird activity. The rest of the videos remained not analyzed.

Figure 1: Nest box in Kelvingrove park and an example of a camera mounted inside the nest box

This problem demonstrates the need for computer vision in ecology. It is becoming increasingly popular as conducting observations is often time-consuming and logistically difficult [5]. In his review of computer vision for animal ecology, Weinstein [5] surveyed 187 existing applications. His findings show that automating observational tasks not only reduces the expenses and time the researchers need to spend analysing the data, but also increases the breadth, duration, and repeatability of ecological studies. Additionally, the use of cameras and automated systems provides a non-invasive way to observe wildlife and avoids the disruption created by direct observations.

1.1 Statement of the problem and approach

Understanding how species adapt to the urban environment, why some species can adapt, why others cannot, and what we can do to ensure that cities are ecologically sustainable and biodiversity-rich are key to determining the impacts

Figure 2: An example from the video footage of a blue tit visiting its nest box carrying an item of food

of urbanisation [3]. The research question motivating this project is whether machine learning can be used to automate the food classification from blue tit nest recordings and aid research in this area as a result.

The ongoing research on urbanization effects conducted at the University of Glasgow started through monitoring the blue tit nest boxes and sought to study blue tits specifically. A crucial part of this is analysing the blue tit nestling diet as it particularly relies on caterpillars. However, the diet investigation is facing the hurdle of manually analysing video footage. Skilled scientists have needed to dedicate hours to trying to identify the types of food brought by adult birds from the recordings. Despite their efforts, the sheer volume of footage taken has made it impossible to work through most of the available data and a new approach is now necessary to move the research forward.

This project focuses on creating a method to classify three basic types of prey: *caterpillars*, *insect*, and *other* food items. Due to the quality of the recordings, the main goal is to identify the proportion of caterpillars fed to the nestlings. To be able to locate and classify the types of food brought by parents to their nestlings, three distinct steps are necessary: Detecting the face of the bird as it enters the nest box, locating facial landmarks such as eyes and beak, and finally detecting and classifying the type of food. Detecting the face will filter out all the frames without the bird. Locating the beak will help with detecting the food as it will narrow down the area of interest. Once the food has been detected it can be classified. If successful, this would remove the manual work currently spent on analysing the footage and give access to a larger sample size by processing a greater volume of data than has been possible by hand. The ability to analyze larger volumes of data will provide more comprehensive insight into why urban blue tits might be struggling.

2. BACKGROUND

This section introduces the urbanization research and presents the studies already done to determine the breeding success of urban blue tits. It explores possible approaches for tackling the bird detection, landmark localization, and food classification problems as outlined in the problem statement.

2.1 Effects of urbanization on blue tits

The blue tit, found throughout Europe and parts of Asia [6], is a commonly seen bird species in city parks and gardens as well as woodland areas. City spaces may offer better nesting spaces and easily accessible food from bird feeders. However, urban dwellers face a different set of problems than their forest counterparts due to the lack of natural prey.

Pollock et al. [1] used integrated behavioural and stable isotope data, obtained from blood samples, to see whether the altered diet of urban and suburban blue tits leads to low breeding success. They assessed arthropod availability, investigated parental provisioning behaviour, inferred diet through stable isotope analysis, and measured the reproductive success of both urban and forest site blue tits.

Caterpillars are a key part of the blue tit nestling diet and other types of food might not be suitable. Their paper made a prediction that the low number of dietary caterpillars would be negatively correlated to breeding success. The proportion of caterpillars provisioned over 30-minute periods decreased from the forest, where this was 82%, to the 54% in the suburban area, to only 27% at the urban site.

The study [1] concludes that the fledgling number per brood was strongly affected by the site. While the number in the forest was 5.06, it dropped to 2.63 in the suburban area, and barely any survived in the urban area with only 0.38. This shows that breeding blue tits lack the optimal food source at urban sites to successfully raise their offspring.

In a second study, Branston et al. [3] compared the effects of urbanization on blue tits and great tits. The data was collected through weekly monitoring of all available nest boxes. It was found that there was a significant mismatch between the peak caterpillar availability and the nestling demand. For forest birds, the peak nestling demand was 5 days before the peak caterpillar availability but for the urban birds, it was 10 days. Urban blue tits had a mean size of clutch 7.8 but the mean clutch size in the forest was 9.3. The caterpillar availability and peak could partly explain the reduced breeding success. This further highlights the need for a system for diet analysis.

Both of the studies collected the data using various methods. While safe for birds, some of the methods do require birds to be caught and handled which creates stress and need greater resources than the computer vision pipeline proposed in the project. Analyzing video recordings is a noninvasive way to obtain important information without disturbing the birds.

2.2 Bird detection

Most of the computer vision bird research is focused on classifying bird species or detecting them in flight for security reasons, such as airport vicinity [7]. This section considers possible methods to detect the blue tit entering the nest box.

Atanbori et al. [8] worked on bird classification from videos. This paper addresses the problem of in-flight classification, which is more suitable for real-life fieldwork than sharp bird images of a certain pose. The suggested set had 320 features, including colour moment, grayscale histogram, curvature scale space, and wing beat frequency. The final set was reduced by two selection strategies: correlation-based and classifier-based. The Random forest classifier achieved

the highest accuracy score of 90% while retaining 80 features from the proposed feature set. The features used in the paper rely on seeing the entire bird and utilizing the colour. Many of them were also dependent on the shape of the flight and its trajectory which is not visible in the blue tit footage. The methods applied in [8] are therefore not transferable to the blue tit project.

Although not directly identifying birds, Ge et al. in [9] utilized You Only Look Once (YOLO) version 5 in combination with Feature Pyramid Network (FPN) to detect bird nests in power lines. Before this research, nest detection was done manually by looking at images collected by drones. Automated bird nest detection is an interesting problem and some similarities to the blue tit project can be found. [9] showed that YOLOv5 is capable of learning some less traditional objects as well, in this case, bird nests. The study achieved 83.4% accuracy in identifying bird nests. It also demonstrated real-time video analysis capabilities by achieving a processing speed of 85.32 frames per second (FPS) on average. The methods in this paper correspond well with the blue tit project, as the goal is to detect the bird as it is entering the nest instead of classifying it.

2.3 Facial landmarks detection

The next important step is to localize the facial landmarks. This will help to estimate the food location in the images. Most of the current research is focused on human faces which is not very useful for birds. The papers in this section describe methods used to learn object or animal landmarks.

Liu and Belhumeur [10] attempted to detect different bird body parts. These included some facial landmarks such as left and right eyes, beak, and forehead. In total, 15 body parts were annotated. First, a classifier made a prediction on what species the bird is, and which one of the 2000 available poses it is making, then the body parts were located in the image. The model's performance was best with 500 pose types for each part, scoring 57.03% on the percentage of correctly estimated parts metric. Although an interesting approach, the blue tit videos do not have all the information used in this method. Only the face is visible when the bird is entering the nest, there is no information on body landmarks or pose. This classifier is likely too complex for the blue tit project which only seeks to predict the eyes and the beak.

Yang et al. [11] selected sheep for facial landmark detection. The researchers used a new feature extraction scheme called triplet-interpolated feature. The paper used a sheep face data set and was able to achieve a good performance despite training on only 500 images. The images in the data set exhibited a wide range of diversity: sheep facial colour and breed, lighting condition, background, occlusion, and head pose. Eight landmarks were manually labeled on each image. 90% of cases were localized with a mean error of less than 10% of the face size. The method was able to deal with occlusion and large pose variation. Especially with the sparse facial landmarks, the method was able to perform better than robust cascaded pose regression and explicit shape regression. The data set used has several similarities to the blue tit images: sparse landmarks and few samples.

Thewlis et al. [12] uses unsupervised learning for object landmark detection utilizing the factorized spatial embeddings method. This is an important problem as it allows for the learning of any object structure while taking the image deformation and viewpoint change into account. Very large, out-of-plane rotations are not considered as these lead to the occlusion of some landmarks. The paper demonstrated that the network was able to learn meaningful landmarks by guessing a warp that will transform landmarks from one image to match the same landmarks in another image. The network learned to do this, not only for human faces but also for cats and shoes. The training required a large amount of data, in the case of cat facial landmarks around 8000 images, and 50 000 images for shoe landmarks. The nest box footage is very low resolution and therefore might lead to problems with the landmarks' accuracy. However, an unsupervised method for learning the landmarks would be preferred as the manual labeling used in [11] can be time-consuming.

2.4 Small object classification

Small object detection remains a difficult problem but the papers in this section present ways to tackle this. This task is made more difficult by having low-resolution images.

Haris et al. in [13] propose a way to deal with low - resolution (LR) images as well as small object detection. Task-Driven Super Resolution (TDSR) uses Super-resolution to map an LR image to a high-resolution (HR) one. This avoids the need to create a detector that is LR-friendly by default. The network used in this paper for super-resolution is Deep Back-Projection Network (DBPN). The detector component is Single Shot MultiBox Detector (SSD).

To train the final combined network the authors calculated the overall loss consisting of a scaled reconstruction loss for the DBPN part and a scaled task loss from SSD. The performance of TDSR was compared to baseline bicubic SR, SR-GAN, and SR-FT+. The results of TDSR exceeded all other super-resolution methods in standard, noisy and blurry images. The average precision of the model was 62.2% when the original HR image was scaled by a factor of 1/4.

While this paper deals exactly with the problems of low resolution and small object detection, to train the super-resolution network HR images are needed as they represent the target (ground truth). The blue tit data set does not have access to such HR images.

Singh et al. [14] explored the fine-grained classification of low-resolution images from 32×32 px to 224×224 px. This paper presents a new technique called attribute-assisted loss to achieve fine-grained classification on low-resolution images. This method gives the network both visual and conceptual attributes, which may not be present in the image, to assist the learning. For example, considering classes in the blue tit data set, 'caterpillar', 'insect', and 'other'; their attributes could be wings and no wings. Or a conceptual attribute such as the environment they live in could be provided.

The model learns from a combined loss which consists of adding the classification loss and the scaled attribute loss. The attributes are no longer required after training. The proposed approach was evaluated on three different networks: ResNet-18, ResNet-50, and DenseNet-121. Each network was trained with three losses: Euclidean distance, Cross Entropy loss, and attribute-assisted loss. The new loss consistently improved the performance of each of the networks. When 32×32 resolution was used the Animals with Attributes-2 data set achieved 53.24% accuracy and the DeepFashion data set had an accuracy of 64.02%. The accuracy grew when higher resolutions were used. The advantage of this method is that it does not require high-

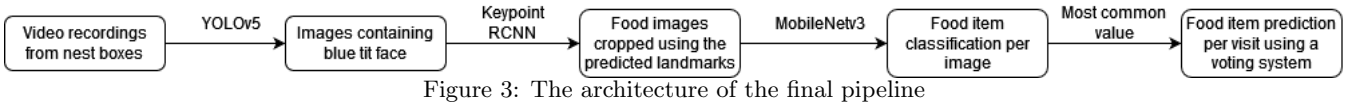


Figure 3: The architecture of the final pipeline

resolution images unlike the super-resolution in [13]. The biggest problem is once again the image availability with only 212 samples compared to thousands of images in Animals with Attributes-2 and DeepFashion.

Rustia et al. in [15] deal specifically with insect classification from low-resolution images found on sticky paper traps. The goal was to automate the classification of insects as a part of integrated pest management. The researchers used a tree-based classifier made up of several image classifier models linked in a tree manner and a Tiny YOLOv3. YOLO was used to detect objects and to crop them out. The model used in the tree structure was a convolutional neural network. The tree is divided into 3 stages. In the first stage, the main target classes are assigned. If an insect belongs to one of the three classes it is then moved to stages 2 and 3 for a more specific classification. Each tree stage was tuned using grid search which resulted in an average F_1 score of 0.94 for the classifier. This is a very good score as the input images were only 128×128 px. The food classification for this project does not need to be so detailed, therefore the top level of this architecture would likely be enough. The main issue is the size of the input images, while [15] uses 128×128 px, the food items are only 48×42 px.

2.5 Summary of findings

Although the blue tit data set is challenging due to its small size, image quality, and specific area of interest, the above findings presented promising solutions. Mainly the use of YOLO in [9] shows that the network can be applied to detect uncommon objects and therefore could be suitable to detect the bird's face. When it comes to locating the facial landmarks, both [11] and [12], could be viable solutions. [11] is supervised learning but specifically deals with sparse landmarks and can be trained even on a small data set. [12] contributed unsupervised learning, which avoids the need to manually label images, but requires a substantially larger data set. This project investigated the suitability of [11] and [12] to decide which approach to follow. The small object classification could be solved by estimating the food location and cropping the picture, then classifying the crop. The use of additional attributes demonstrated in [14] could greatly improve the chances of learning the food classification. The main advantage of this system is that it does not require high-resolution images like in [13].

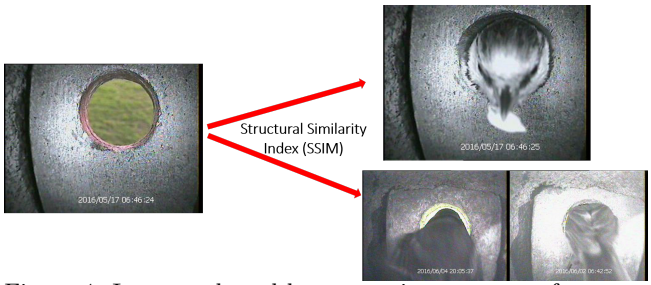


Figure 4: Images selected by comparing an empty frame to all the frames in a video

3. IMPLEMENTATION

The following sections describe the data available for this project and explain the implementation of each separate component of the pipeline.

3.1 The data set

The footage was taken in 2016 in selected blue tit nest boxes in the Glasgow urban gradient [2]. There are around 500 hours of footage available. The videos are between $1-30$ minutes long and the visits constitute only a very small part of each video. The frame rate is 12 frames/s and the size of each video is 320×240 px.

The original research for which these recordings were made used only 5.5 hours to get an overview of food types fed to the nestlings. There were four main class types: caterpillar, insect, other, and unidentified. In total, there were 267, out of which 212 (79%) are identified. The insect class is further divided into three sub-classes: Diptera, Hemiptera, and Arachnida. For this project, the sub-classes were not considered by the classifier. The data sets needed for the blue tit detection and the landmark localization were to be constructed from scratch. Roboflow [16] and COCO Annotator [17] were used for labelling the images.

3.2 Pipeline overview

Figure 3 shows the overall architecture of the pipeline. Multiple models needed to be assembled to be able to detect the visits and classify each food item. Starting from a video, YOLOv5 selects relevant frames and saves them as images. It also divides the images into visits, the parents can come to the next box multiple times per video. These images are then passed through the Keypoint RCNN to find the eyes and the beak. Finally, the frames are cropped using the beak location to only include the food. These crops are classified using MobileNetv3. If there are multiple predictions per visit, the most commonly predicted class is selected as the food item delivered during the visit. The following sections provide a detailed description of the implementation of each of the components of the pipeline.

3.3 Bird face detection

This section describes the approaches implemented to filter out the empty frames and save the images of birds entering the nest box.

3.3.1 Naïve approach

After examining the videos a clear pattern emerged. Most of the videos are made up of empty frames - the nest box entrance when nothing is happening. When the bird enters the nest the hole is covered by it. Therefore, removing the empty frames should result in retaining only useful information. To achieve this, a simple Python script would save an image of an empty entrance and then compare every other frame in the video using Structural Similarity Index (SSIM). If the score was below 0.9, the frame would be saved. However, this led to images of both the bird entering and leaving the nest being saved as shown in figure 4. The images show-

(a) 'Exp' model - correct selection

(b) 'Exp+noise+blur' model - correct selection

(c) 'Exp' model - correctly ignored

(d) 'Exp+noise+blur' model - incorrect selection

Figure 5: Examples of two different YOLO models selecting frames from a video

ing the tail of the bird do not have any useful information and would only create confusion in the next stages.

This simple system was still useful for saving a large number of images from the videos. It was easy to separate the target images this way so that they could be labelled to train a more robust and reliable bird detection model.

3.3.2 YOLOv5

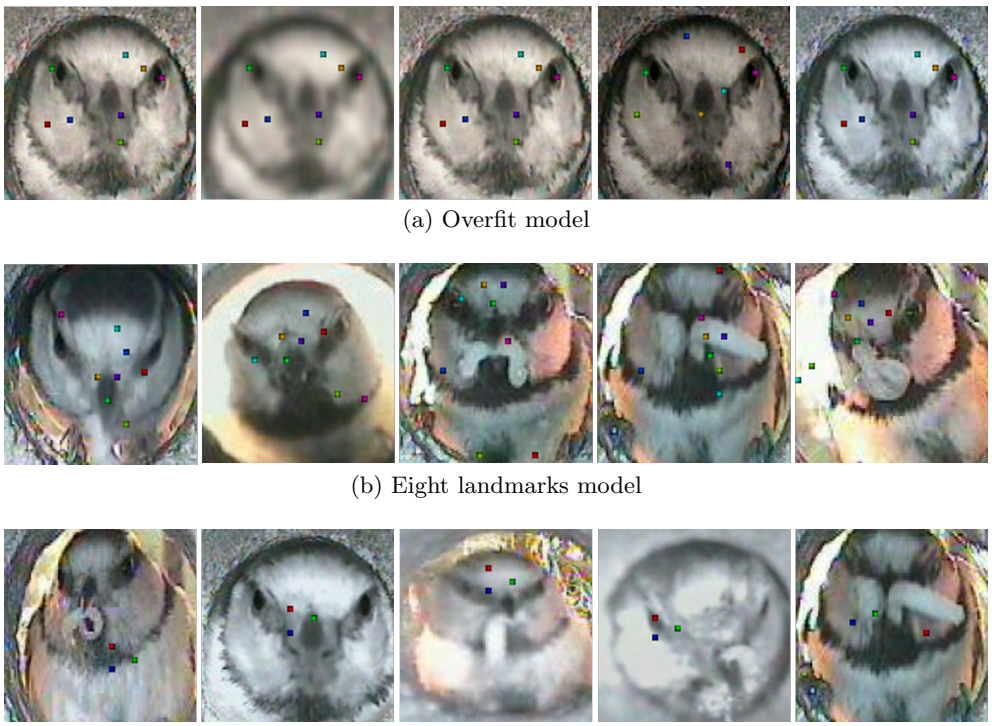
YOLOv5 was used to detect a bird's face in the videos. This approach was inspired by [9] where the researchers used YOLO to detect less common objects - nests.

YOLO network differs from other object detection networks. It predicts the bounding boxes and the class probabilities at the same time. Instead of repurposing a classifier for object detection, YOLO views it as a regression problem. The original architecture is very simple: 24 convolutional layers followed by two fully connected layers. Furthermore, YOLO is much faster than traditional object detection algorithms. It can achieve 45 FPS [18]. The later versions of YOLO, namely YOLOv4 and YOLOv5, improve mean average precision and increase the speed to 65 FPS [19].

For the blue tit detection task, the baseline data set had 249 training images, the validation set consisted of 55 images, and the testing set contained 53 images. To train different models, images in the training set were augmented to increase the number of training examples. The first data set varied in exposure only, and the second data set had augmented exposure, levels of blur, and added random noise. Each of the models was trained over 100 epochs.

The detect.py script, which is used to run the trained model, was modified to save the images where the blue tit was detected without the predicted bounding box. Only the saved pictures are used in the later stages, the videos can be disregarded. Examples of correct and incorrect blue tit detections are shown in figure 5 and details of these models are given in section 4.1.

This part of the pipeline is also responsible for tracking and separating the visits. When a blue tit is detected, the



(a) Overfit model

(b) Eight landmarks model

(c) Three landmarks model

Figure 6: Factorized spatial embeddings landmark predictions

frame number is stored in a dictionary along with the visit number. The visit number is increased when the difference between the current frame and the last selected frame is at least 12. This means that it should be safe to assume if there are no detections for at least one second, the next detection can be considered a new visit. The number of frames was increased from the original five which turned out to be too sensitive and detecting too many visits. The dictionary is saved as a pickle file to be loaded later during the food classification phase.

3.4 Facial landmarks detection

For facial landmarks detection, both unsupervised and supervised learning was attempted. The unsupervised method was originally preferred as it does not require the data to be labelled manually.

3.4.1 Factorized spatial embeddings

This model used the network provided in [12] to try and learn the facial landmarks in an unsupervised way. To start the training process the network only needs to know how many landmarks it should predict. The position of the landmarks is not pre-defined, they are selected randomly during training.

The above method can be described as learning a function that maps object points to the corresponding pixels in the image. It needs to find a warp that when applied to the original object points will be consistent with the change of the viewpoint. A neural network is supplied with pairs of images of the same object but since the warp is unknown the network tries to guess it randomly and then applies it to the first picture to see if it transformed it into the second picture.

The code [20] required a major Tensorflow version upgrade and some adjustments for Google Colab compatibility. In the first experiment, it was verified that the network still works after the upgrade and that this type of image is not completely unsuitable for the task. The model was overfit using the same image augmented in different ways. Figure 6 a) shows the model's prediction when the same image with different augmentations was used to train it. In total 12 augmented images were used for training. The landmarks are consistent across most of the predictions. The landmarks matched in 4/5 cases therefore it was safe to move ahead with this approach to try training with more data.

To increase the images in the data set Albumentations [21] augmentations were used. The transform included varying contrast, blur, and noise. This method then generated new images rather than replacing the existing ones as is the case with Pytorch transforms. At first, the entire frames, see figure 2, were passed to the network but this resulted in the majority of the landmarks being predicted outside the face. To deal with this problem, the frames selected by YOLOv5, as shown in figure 5, were saved as crops of the head based on the predicted bounding boxes. This ensured that the points selected were located on the face.

Figure 6 b) shows predictions of 8 landmarks from a network trained on a set containing 4000 images. The blue tit data sets ranged from 2000 images to 6000 images. However, regardless of the number of images used no improvements were detected. There appears to be no consistency in the predictions and the predicted points do not seem to select the same areas in the images. There might be multiple

Figure 7: Example of a food item cropped out for classification.

causes to why this approach did not work. The authors of [12] state that the network cannot deal with landmark occlusions yet but there are multiple examples in the blue tit data set where one of the eyes is not visible or the beak is covered by the food item. Another possible explanation could be that too many augmentations were used which contributed to an already difficult data set due to the low quality of the images. Some of the augmentations could be unsuitable such as image rotation or horizontal flip. Finally, there is no way to tell the predicted landmarks apart and another model likely would have been needed to approximate which of the predicted points correspond to beak and eyes.

As the last attempt, the network was trained to predict three landmarks instead of eight. The results can be seen in figure 6 c). The same problem as before occurs, there is no consistency and the landmarks are not selecting anything useful.

3.4.2 Keypoint RCNN Pytorch

The second approach utilized supervised learning. 550 images were labelled using the COCO annotator to mark eye and beak locations. Out of this set, 490 images were then used to train a keypoint region-based convolutional neural network (RCNN) by Pytorch which is based on Mask R-CNN created by He et al. [22]. The remaining 60 images were used for testing.

Mask RCNN is a network that combines two tasks: object detection and semantic segmentation. Two models are used to achieve this Faster RCNN and Fully Convolutional Network (FCN). Faster RCNN is responsible for selecting the area of interest marked by a bonding box. The segmentation part is done by the FCN where each pixel is assigned a class to create an object mask.

The network was initialized with pre-trained weights for the ResNet50 backbone and trained over 20 epochs to detect three key points. There were multiple predictions per image containing a bounding box for the head and a set of three landmarks for the eyes and the beak. The prediction with the highest confidence score was chosen.

3.5 Food classification

The final and most important model is the food classifier. MobileNetv3 [23] was selected for this task. This was because it maintains a balance between the system requirements needed for training while displaying good performance on image-based tasks. Furthermore, its architecture is not as complex as some of the other most current deep learning models and allows for more freedom when adding and changing layers if needed.

MobileNetv3 has multiple improvements compared to its predecessors. This version adds squeeze and excitation blocks which help find the areas the network should pay the most attention to. MobileNetv3 also introduces a new activation function Hard Sigmoid and adds a swish function. This helps to update dead neurons instead of never changing their values.

MobileNet is constructed from a series of convolutional blocks, or in the case of MobileNetv3, bottlenecks. Each block consists of multiple convolution blocks and some of the bottleneck blocks include squeeze and excitation block (SE block). Convolution block consists of the convolution function and batch normalization function which can be either Hardswish or ReLu. The SE block consists of a pooling function, two fully connected layers, ReLu, and Hard sigmoid. [23] describes the order of these blocks, input and output channels, kernel size, and whether SE block is used in each bottleneck. There are two versions of MobileNetv3: large, targeted at high-resolution images, and small, used for low-resolution images. However, in this project, the large structure performed better than the small structure.

The image input for this network is in the form of height (H) × width (W) × channels (CH). Since the images are loaded with the cv2 library, the array needs to be permuted to match the tensor input. The images are not converted to grayscale for input.

The images of blue tit selected by the previous stages were cropped using the predicted beak location. The beak point was used as the center of the upper edge of the rectangle. The bounding boxes were used to estimate the optimal size of the crop. This was calculated as the average width = 48 px and height = 42 px. An example of this crop can be seen in figure 7. The other crop size considered was the longest width = 130 px and height = 70 px. However, this added too much extra noise around the food item and led to very poor performance.

The food data set is extremely unbalanced. Out of 463 samples (after augmentation), there are 278 'insect' images, 138 'caterpillar' images, and only 47 'other' images. To address this, different sampling strategies were tried during the testing phase. For the first attempt, the images were distributed randomly into training, validation, and testing sets. These subsets would then also be unbalanced. A random weighted sampler was used for the training set data loader, assigning the largest weight to the 'other' class, then the 'caterpillar', and the 'insect' class had the smallest weight.

The second attempt had validation and testing sets balanced from the beginning. Each class was represented by a similar number of images. The training class was sampled using a balanced batch sampler [24] so that each training mini-batch would include the same number of examples for each class.

3.6 The full pipeline

The full pipeline, as shown in figure 3, is assembled into a single Google Colab notebook. The pre-trained weights for all three models are downloaded from cloud storage, all other necessary repositories are cloned from Github. The videos can be uploaded to the notebook's temporary storage or Google Drive.

The pipeline begins by processing a video into separate frames. While detecting the blue tit face in a video it also tracks which visit it belongs to. When the video is processed all the selected images are saved in a folder along with a pickle file containing the dictionary of frames and visits.

The next step takes the predicted frames and tries to locate the landmarks. If any landmarks were predicted it checks if one of them is a beak. This is done by comparing the predicted y coordinates to each other. If the difference between these is less than five, it is assumed that one of the eyes was predicted twice. In this case, one of the coordinates gets shifted to where the beak should be by adding the average distance between the eyes and the beak. This distance was calculated from the already labelled data. Although this is a naïve method, it improved the training and validation losses. After this step, the food crop is taken using the beak location and the average bounding box size for the food items estimated in the earlier steps. The food crops are saved in a separate folder to allow for extra control and verification between the steps.

The third model uses the food crops to predict the food item for each frame. The dictionary containing the visit number and the image is updated with the predicted labels. Finally, a voting system is used to determine what type of food was brought during a particular visit. The most detected class from images belonging to the same visit is selected. After this, a summary of visits is printed, stating the visit number, the time of the visit in the video, and the food class.

4. EVALUATION

This section presents the results for each of the models used in the pipeline. The Google Colab GPU was used for running all of the components.

4.1 Bird detection

The videos are very fast to process. A video of length 1.5 minutes took only 14 seconds to process. The largest available 30-minute video took 4 minutes and 26 seconds.

Three different models were evaluated by plotting the ROC curve, calculating the average intersection over union (IoU) value, and assessing the predictions with a confusion matrix. For simplicity, the models are called based on what augmentations were used on the training data: 'no aug' for the baseline data set with 249 images, 'exp' for the data set augmented with exposure containing 720 images, and 'exp+noise+blur' for the set with exposure, noise and blur augmentations containing 747 images.

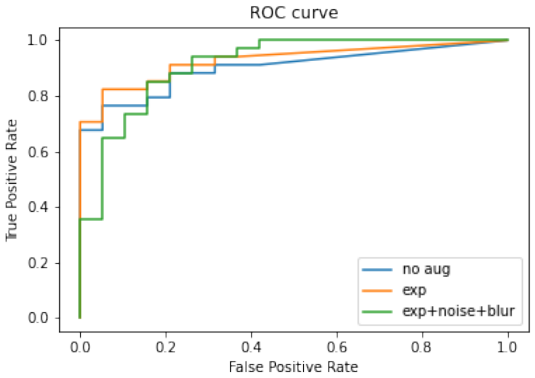
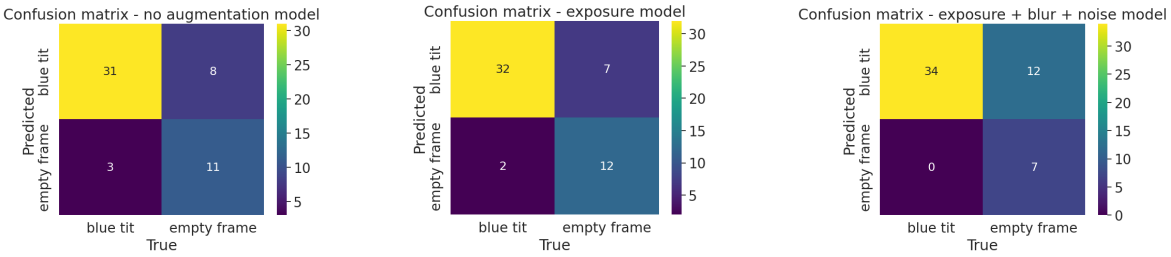


Figure 8: ROC curve for the three selected models



(a) Predictions from 'no aug' model

(b) Predictions from 'exp' model

(c) Predictions from 'exp+noise+blur' model

Figure 9: Comparison of three YOLOv5 models on the test data set. Model b) was selected as it maintains high true positive and negative rates while minimizing false positive predictions.

Figure 8 shows the ROC curve for each of the three constructed models. The 'exp' model is the closest one to the top left corner. The confidence threshold for the bird detection was selected to be 0.4. That is any detection with a confidence value greater than 0.4 will be saved and used in the later stages of the pipeline.

IoU value of 0.7 or greater is considered as good performance [25]. The average IoU for the 'no aug' model was 0.62 which is below the threshold. The 'exp' model achieved the highest IoU with 0.75. Finally, the 'exp+noise+blur' model scored 0.7 just passing the threshold.

The predictions from each model can be seen in figure 9. The true positive and true negative values between all three models are very close but the 'exp' model has the edge with the highest IoU score and the smallest number of false positives. The confusion matrix in figure 9 c) shows that the 'exp+noise+blur' model achieved the highest number of correctly selected blue tits, however, it also selected a large number of frames that are not of interest. This is a serious disadvantage as images that do not contain the face could confuse the pipeline in the later stages. Furthermore, these images had a confidence value similar to the true positives and would be hard to filter out automatically.

Based on the ROC curve, IoU value, and predictions displayed in the confusion matrix, the 'exp' model was selected to be used in the final pipeline.

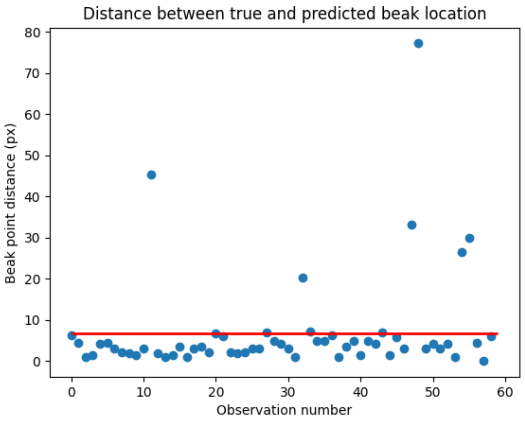


Figure 10: Average beak distance between true and predicted locations. The majority of the points are below average.

4.2 Facial landmarks detection

About 5−7% of images are lost between the bird detection step and the landmark predictions. This is mainly due to wrong images being selected by the YOLO network which are then subsequently ignored for the landmark prediction.

The average head bounding box in the landmarks data set is 97 × 88 px. This means that the head makes up around 11% of the entire image. To assess the performance of the landmark detector, the average distance between the true points and the predicted points was calculated over 60 predictions. 490 images were used to train this model. The error is calculated as the fraction of the distance between the label and prediction over the head diameter (88 px).

The average distance between the real landmarks and the predicted landmark to the left eye was 7 px which is equal to 8% error, to the right eye = 4 px with error 5%, and to the beak = 7 px equating to 8% error. The average beak distances per prediction are shown in figure 10. Examples of predictions can be seen in figure 11.

4.3 Food classification

There are two main models: one trained to predict three classes: caterpillar, insect, and other; with 399 training images after augmentation, and the other to predict only two classes: caterpillar and insect; with 392 training images after augmentation. The three-class model achieved 58.8% accuracy on 24 testing images. The classifications are shown in figure 12 a). The data set used was extremely unbalanced, therefore the majority of examples in training, validation, and testing subsets consist of the 'insect' class. This heavily influences the performance. The 'other' is only predicted correctly once, and there is a lot of confusion with the 'caterpillar' class. The model tends to predict the 'insect' class in the majority of the cases as it was trained on a set where more than half of the examples were of this class.

Since the 'other' class is so underrepresented the second model only predicts 'caterpillar' and 'insect'. While it would be interesting to have more detailed predictions of the food items, the primary goal of this project is to find out the portion of the caterpillars fed to the nestlings. This means that the caterpillar accuracy was prioritized over smaller class identification. The validation and testing subsets were balanced to have a similar number of examples for both classes and the training batches included the same amount of examples per class. These changes lead to a performance increase to 69% accuracy. Figure 12 b) shows that the 'caterpillar' class now gets predicted more reliably.

(a) Correctly predicted landmarks

(b) Correctly predicted landmarks

(c) Incorrect landmark prediction, the model does not handle occlusions well

Figure 11: Keypoint RCNN model predictions (yellow) and original labels (red)

4.4 The pipeline

It takes 1.5 minutes to run the entire pipeline on a 3.5-minute video. The goal of this pipeline is to be user-friendly, everything can be set up using a single Colab notebook [1]. A zip file with all the models and altered code is downloaded into the user's Google Drive. The repositories to run each model are cloned from GitHub and saved. The setup portion therefore only needs to be executed for the very first time. The user can then upload desired videos into a video folder in Drive and run the pipeline code without requiring specific knowledge of the code. However, the pipeline displays all the code for detection and classification, and the user is able to change any parameters, such as the YOLO detection confidence, freely.

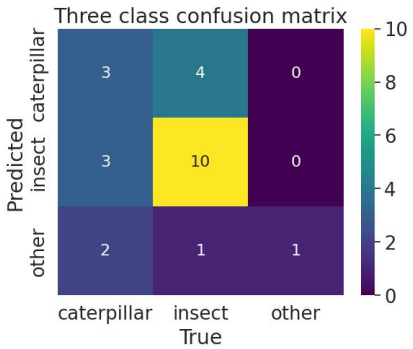
Figure 13 a) shows probabilities per image from a single visit. There were 17 frames in total and the food item was predicted correctly as a caterpillar. The majority (10) of the images were classified as 'caterpillar' but some of the images (7) were detected as the 'insect' class. The voting system finds the right classification by taking all visit predictions into consideration. The images in figure 13 b)-g) show examples of frames that were classified as 'caterpillar' or 'insect' with the corresponding probability scores.

Figure 14 shows an overview of the pipeline performance from a single video analysis. Part a) of figure 14 displays the probabilities of insect and caterpillar classes per image that was detected by the YOLO model and cropped by the keypoint RCNN network. Four visits were detected in total and they are separated by the dashed lines. The number of images per visit is highly variable, between 5 and 32 frames. There are only three visits in this video and all should be classified as caterpillars. However, the last visit was split into two by the pipeline when no bird was detected for more than 12 frames. This was caused by the bird's hesitation to enter the next box resulting in large head pose variations where some angles were not detected.

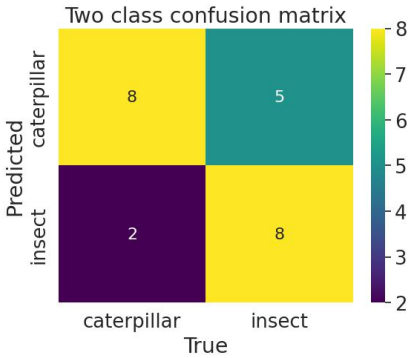
Part b) in the figure 14 gives the total number of votes per class per visit. The vote differences are not very high between each visit which suggests the food classification model could benefit from more training data.

It can be seen in figure 14 a) visit 1 there are sections of insect and caterpillar detections alternating. An example of wrong classification for frames $85-97$ is given in figure 14 c). The food crop is selecting the wrong part of the image leading to a wrong classification. All of the frames in this section had a similar issue. This might have happened because of the head rotation which led to wrong landmark detection. Figure 14 d) gives an example for frames $98-113$ which were correctly classified as caterpillars. The bounding box is in the right place in these images. The wrong food crop repeats again in frames $114-122$. This led to visit 1 being wrongly classified as 'insect' when it should have been 'caterpillar' as in visits $2-4$.

[1] All the code sources and the pipeline itself can be found in this notebook.



(a) Model predictions for three classes



(b) Model predictions for two classes

Figure 12: Comparison of two MobileNet v3 models performance. Model b) is used in the pipeline as it is more accurate with the caterpillar prediction.

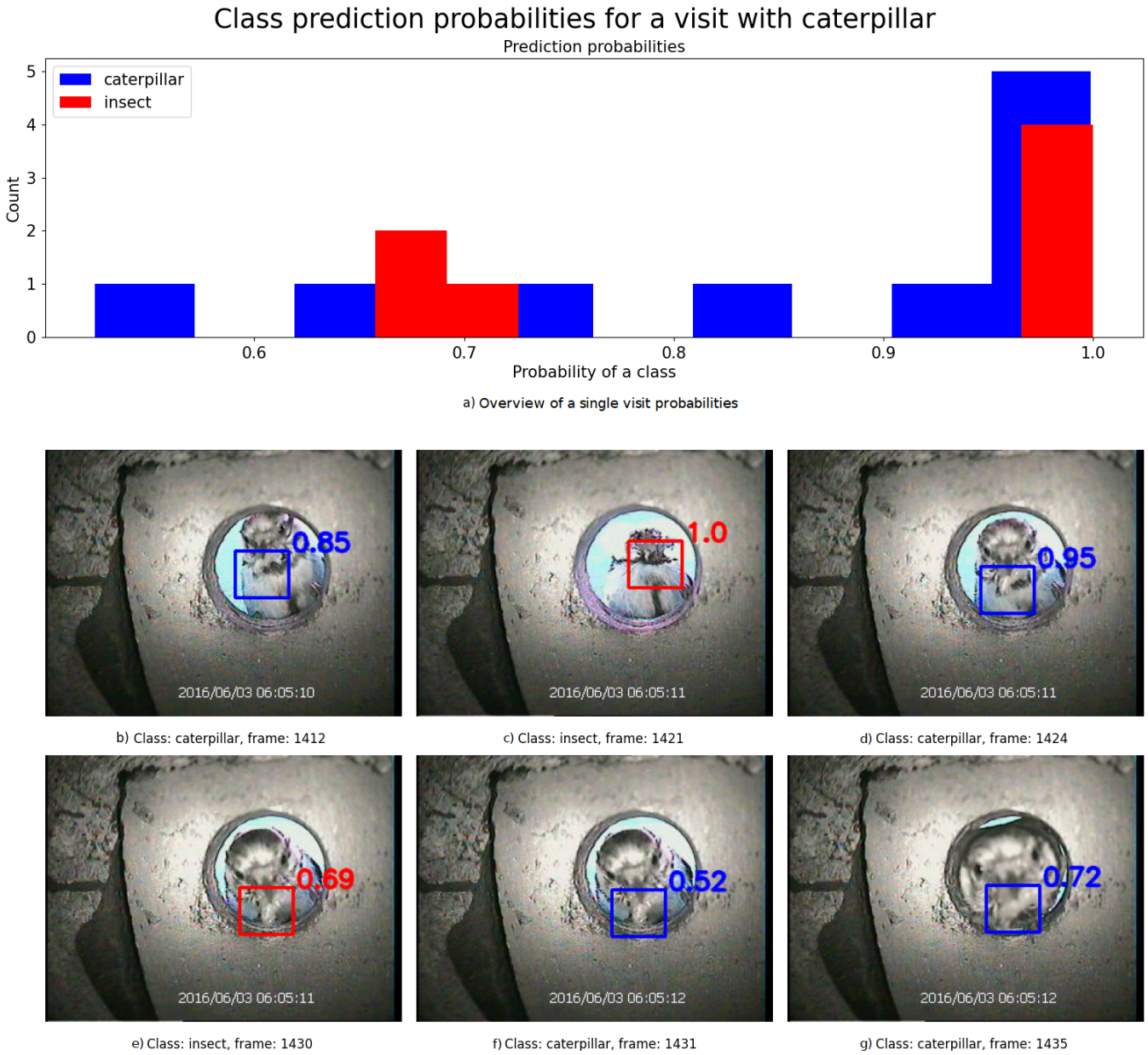
5. DISCUSSION

Despite using only a few hundred images for each task, the deep learning methods were able to achieve a surprisingly good performance. The selection of deep learning methods was driven by the literature survey as convolutional neural networks have recently had great success when it comes to image processing.

A large portion of this project was dedicated to constructing brand-new data sets. There is a lot of variability in the videos: the brightness changes drastically between each of the nest boxes, the entrance is in different locations, and there are many different head poses made by the birds. It was difficult to create sets that would be well-balanced and cover most of the conditions that appear in the videos. Specifically, the bird detection data set had to be modified multiple times.

The landmark model cannot differentiate between visible and not visible landmarks. This should not pose a big problem, as long as the eyes and the beak are being predicted it should not matter if they are labelled as visible or not. The main issue comes from images where the head is visible but the food is not. Here the bird detector correctly saves the frame but there is no useful information in it. Some head poses also lead to wrong predictions. Adding these to the training set could help solve the problem.

The food classifier struggled to decrease the validation loss. It would decrease the first 10 epochs but then it would fluctuate between the values of 1-2 during the rest of the training. This might be because of the lack of data needed to train a more accurate model in this particular method. The



a) Overview of a single visit probabilities

b) Class: caterpillar, frame: 1412

c) Class: insect, frame: 1421

d) Class: caterpillar, frame: 1424

e) Class: insect, frame: 1430

f) Class: caterpillar, frame: 1431

g) Class: caterpillar, frame: 1435

Figure 13: Final pipeline single visit probabilities. Caterpillar food class was correctly selected.

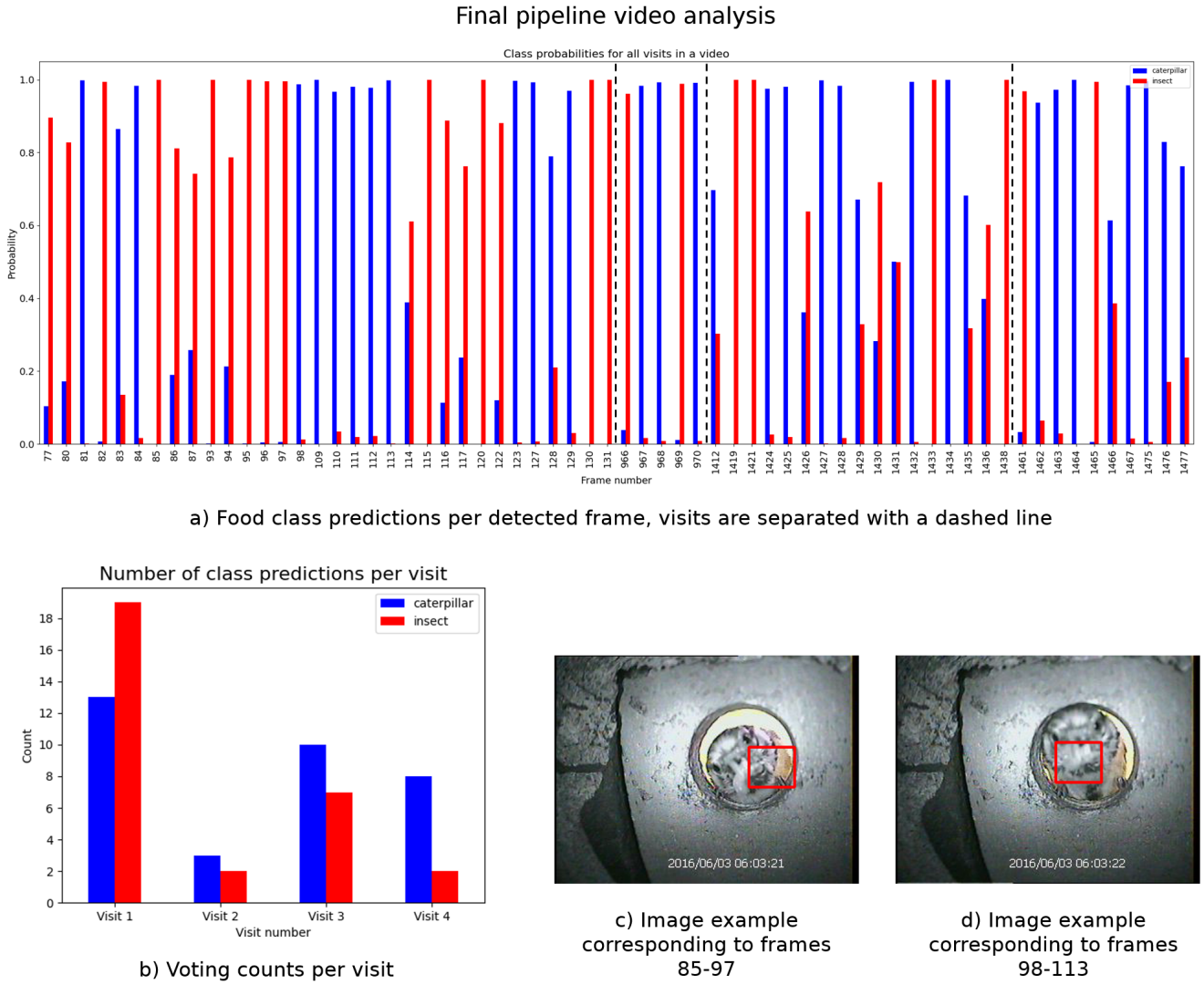
training loss was decreasing throughout the entire training, stopping around the value of 0.002.

For the original method of manual diet classification used in [2] it was necessary for the scientists to watch the videos and find all the nest box visits. This resulted in hours of extra work. The manual method remains more accurate at 79% identified food items while the automated pipeline was only able to achieve 69% on a small test set. However, it is able to process all 5.5 hours of analyzed videos in around 45 minutes and the results can be easily verified by looking at the saved images from each video. With more data, the classifier could improve its accuracy so that it is similar to the manual method.

5.1 Challenges

This project was challenging due to the low quality of the videos. The images had a very poor resolution with only 320×240 px. This is very low for a task that is concerned with small object classification. The other problem was the small quantity of expertly labelled food data. The labels were provided by the biodiversity department and contained less than 300 samples. While selecting head, eyes, and beak in the images is something that can be done by anyone, adding more data to the food item data set requires strong insect identification skills. This data was also extremely unbalanced, the 'other' class made up only 5.6% of all the labels. This is one of the reasons why only 23 images were used for testing in the food classification section. The most common division is 80% for training and 20% for testing. With deep learning approaches requiring large quantities of data, I opted to use most of the available images for training to have a chance of learning a good model. Although 500 hours of videos were available, the supervised learning methods made it difficult to utilize most of them in the case of YOLO and the landmark model. This is because the labelling had to be done manually from scratch which creates hours of extra work. In the future, this problem could be solved by using the current models to generate more training data and then training new models.

Since this project was completed fully by using Google



a) Food class predictions per detected frame, visits are separated with a dashed line

b) Voting counts per visit

c) Image example corresponding to frames 85-97

d) Image example corresponding to frames 98-113

Figure 14: Prediction analysis of a single video processed by the pipeline. Visits $2-4$ have correct caterpillar predictions. The insect class was incorrectly predicted for visit 1, caused by wrong food item crops.

Colab, all the data processing had to rely on Google Drive for storage. However, it turned out Drive is not very suitable for video and image processing. The free version has a very limited capacity - only 15 GB. Uploading large quantities of videos had to be done in batches. The batch needed to be processed into frames and the videos were then deleted to make space for the next batch. The number of images being saved at once also posed a problem, oftentimes multiple images would not save at all. Manipulating hundreds of images led to the Drive freezing and crashing the browser. This method was good for experimentation and development but for serious processing of large quantities of data and subsequent analysis, it would be better to run this code locally. The ability to run this code directly on a computer relies on access to a powerful graphics card which most users might not have.

There are definitely positives to using Google Colab and Drive. Apart from being free, they can be accessed instantly and most of the libraries are pre-installed. Using the Colab notebooks themselves was not a problem, most of the issues stemmed from the use of Google Drive.

6. CONCLUSIONS AND FUTURE WORK

This project created a novel method for automated blue tit nestling classification which has been done manually up until now. As far as I am aware, there are no automated methods to analyze bird nestling's diet. The food classifier accuracy achieved 69% but this system has different benefits too. The first two steps of the pipeline are capable of finding and saving the nest box visits. This already removes a large chunk of manual work because anyone interested in analyzing the blue tit nestling's diet can simply look at a handful of images instead of searching through a 30-minute video with an unknown number of visits which often last only half a second each.

6.1 Future work

It would be beneficial to try out different approaches for food classification. Deep learning has gained great popularity and convolutional neural networks tend to have a very good performance on image data but with roughly 300 images available for this task it might not be the best way to train a classifier. The next best candidate for this task is currently the scale-invariant feature transform (SIFT).

The blue tit behaviour is something to consider as well. Occasionally, the blue tit is hesitant to enter the nest box. This could be because there is a threat or it simply got disturbed. The pipeline is currently unable to detect the cases where the bird arrived at the box but left without feeding the nestlings. A case of this was shown in figure 14 when the bird was extremely hesitant, generating a large number of frames per visit and leading to a case where a single visit was detected as two separate ones.

The results show that with more data, this pipeline has the potential to be fine-tuned to achieve higher reliability in food detection and classification. It is already capable of speeding up the work currently needed to analyze the videos. All the methods explored in the project could provide good building blocks for future ecology computer vision tasks.

Acknowledgments. I would like to thank my supervisor Dr Nicolas Pugeault for his guidance during this project and helpful ideas. I would also like to thank Dr Davide Dominoni and Dr Pablo Capilla Lasheras for not only providing the data for the project but also offering useful insight into urbanization research. Finally, I would like to thank my boyfriend Lennart for supporting me throughout my entire time at the university.

7. REFERENCES

[1] C.J. Pollock et al. Integrated behavioural and stable isotope data reveal altered diet linked to low breeding success in urban-dwelling blue tits (cyanistes caeruleus). *Sci Rep 7*, 7(1), 2017.

[2] Jarrett C. et al. Bitter fruits of hard labour: diet metabarcoding and telemetry reveal that urban songbirds travel further for lower-quality food. *Oecologia*, 193(2), 2020.

[3] C.J. Branston et al. Urbanisation weakens selection on the timing of breeding and clutch size in blue tits but not in great tits. *Behavioral Ecology and Sociobiology*, 75(11), 2021.

[4] One Health &; Veterinary Medicine School of Biodiversity. Urban-rural gradient study system. https://www.gla.ac.uk/schools/bohvm/research/about/researchstudysystems/urban-ruralgradientsystem/, last accessed on 11/04/23.

[5] B. G. Weinstein. A computer vision for animal ecology. *Journal of Animal Ecology*, 87(3):533–545, 2018.

[6] The Cornell Lab. Eurasian blue tit - ebird, 2022. https://ebird.org/species/blutit/, last accessed on 27/03/23.

[7] Shakeri et al. Real-time bird detection based on background subtraction. In *Proceedings of the 10th World Congress on Intelligent Control and Automation*, pages 4507–4510, 2012.

[8] J. Atanbori et al. Classification of bird species from video using appearance and motion features. *Ecological Informatics*, 48:12–23, 2018.

[9] Z. Ge et al. Bird's nest detection algorithm for transmission lines based on deep learning. In *2022 3rd International Conference on Computer Vision, Image and Deep Learning I& International Conference on Computer Engineering and Applications (CVIDL I& ICCEA)*, pages 417–420, 2022.

[10] J. Liu and P. N. Belhumeur. Bird part localization using exemplar-based models with enforced pose and subcategory consistency. In *2013 IEEE International Conference on Computer Vision*, pages 2520–2527, 2013.

[11] H. Yang et al. Human and sheep facial landmarks localisation by triplet interpolated features. *CoRR*, abs/1509.04954, 2015.

[12] J. Thewlis et al. Unsupervised learning of object landmarks by factorized spatial embeddings, 2017.

[13] M. Haris et al. Task-driven super resolution: Object detection in low-resolution images. In *Neural Information Processing*, pages 387–395. Springer International Publishing, 2021.

[14] M. Singh et al. Enhancing fine-grained classification for low resolution images. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2021.

[15] D. J. Rustia et al. Tree-based deep convolutional neural network for hierarchical identification of low-resolution insect images. January 2021.
[16] B. Dwyer et al. Roboflow, 2022. Available from https://roboflow.com.
[17] J. Brooks. COCO Annotator, 2019. Available from https://github.com/jsbroks/coco-annotator/.
[18] J. Redmon et al. You only look once: Unified, real-time object detection, 2016.
[19] A. Bochkovskiy et al. Yolov4: Optimal speed and accuracy of object detection, 2020.
[20] A. Dabouei. Factorized-spatial-embeddings, 2019. Available from https://github.com/alldbi/Factorized-Spatial-Embeddings.
[21] A. Buslaev et al. Albumentations: Fast and flexible image augmentations. *Information*, 11(2), 2020.
[22] He Kaiming et al. Mask r-cnn, 2018.
[23] A. Howard et al. Searching for mobilenetv3, 2019.
[24] F. Galatolo. pytorch-balanced-batch, Mar 2023. https://github.com/galatolofederico/pytorch-balanced-batch, last accessed on 03/04/23.
[25] Hasty.ai. Intersection over union (iou), Mar 2022. https://hasty.ai/docs/mp-wiki/metrics/iou-intersection-over-union, last accessed on 05/04/23.