Automating Bird Detection Based on Webcam Captured Images using Deep Learning

Alex Mirugwe¹, Juwa Nyirenda², and Emmanuel Dufourq³,⁴,⁵

¹ University of Cape Town, Cape Town, South Africa  
mirugwealex@gmail.com
² University of Cape Town, Cape Town, South Africa  
Juwa.Nyirenda@uct.ac.za
³ Stellenbosch University, Stellenbosch, South Africa
⁴ African Institute for Mathematical Sciences, Cape Town, South Africa
⁵ National Institute for Theoretical and Computational Sciences, South Africa  
edufourq@gmail.com

Abstract

One of the most challenging problems faced by ecologists and other biological researchers today is to analyze the massive amounts of data being collected by advanced monitoring systems like camera traps, wireless sensor networks, high-frequency radio trackers, global positioning systems, and satellite tracking systems being used today. It has become expensive, laborious, and time-consuming to analyze this huge data using manual and traditional statistical techniques. Recent developments in the deep learning field are showing promising results towards automating the analysis of these extremely large datasets. The primary objective of this study was to test the capabilities of the state-of-the-art deep learning architectures to detect birds in the webcam captured images. A total of 10592 images were collected for this study from the Cornell Lab of Ornithology live stream feeds situated in six unique locations in United States, Ecuador, New Zealand, and Panama. To achieve the main objective of the study, we studied and evaluated two convolutional neural network object detection meta-architectures, single-shot detector (SSD) and Faster R-CNN in combination with MobileNet-V2, ResNet50, ResNet101, ResNet152, and Inception ResNet-V2 feature extractors. Through transfer learning, all the models were initialized using weights pre-trained on the MS COCO (Microsoft Common Objects in Context) dataset provided by TensorFlow 2 object detection API. The Faster R-CNN model coupled with ResNet152 outperformed all other models with a mean average precision of 92.3%. However, the SSD model with the MobileNet-V2 feature extraction network achieved the lowest inference time (110ms) and the smallest memory capacity (30.5MB) compared to its counterparts. The outstanding results achieved in this study confirm that deep learning-based algorithms are capable of detecting birds of different sizes in different environments and the best model could potentially help ecologists in monitoring and identifying birds from other species.

Keywords: Deep Learning, Transfer Learning, Single-Shot Detector, Faster R-CNN, ResNets

1 Introduction

In the world of ecosystem preservation, domestic and wildlife animal monitoring and identification are very important areas of research as they help ecologists and conservation practitioners to monitor different species especially animals on the verge of extinction, understand species abundances and the effect of the environment on wildlife [15], and finally, formulate conversation and management policies [27]. More than ever, improvement in animal monitoring systems\methods are needed if we are to preserve the existing wildlife from the increasing threat of climate change, human-animal poaching, resource acquisition, and endangered species [63, 50, 60].

Due to the inefficiency of the traditional wildlife monitoring techniques, several modern tools have been developed such as motion sensor camera traps [30], wireless sensor networks [3, 57], high-frequency radio trackers [16], and the global positioning system (GPS) and satellite tracking systems [17]. These observational technological advancements have enhanced the ability to obtain massive, long-term, cross-scale, and heterogeneous data [18]. It is also helping ecologists to document all aspects of wildlife like feeding, movement, sleeping, and interaction with one another, something hard to be done through physical human monitoring [48]. In some cases, it is dangerous or even impossible for humans to physically monitor some wild (e.g predators) and sea species. For example, in 2017, a wildlife ecologist known as Krisztian Gyongyi was attacked and killed by a rhino while he was tracking animals in Akagera National Park in Rwanda\footnote{For more information about the story: https://news.mongabay.com/2017}. Therefore using automated tools can be very helpful to collect data on such animals.

But these new wildlife monitoring technologies have resulted in huge data sets that have greatly outpaced the traditional manual analytical techniques as they are costly, labour intensive, and time-consuming [53, 47, 2]. Even the traditional machine learning tools like Support Vector Machine (SVM), random forest, Linear, Discriminant Analysis (LDA), K-nearest neighbor (KNN), and Principal Component Analysis (PCA) are not suitable because they quickly saturate whenever the data volume increases [44]. For example, the 225 camera traps deployed by Snapshot Serengeti\footnote{https://www.zooniverse.org/projects/zooniverse/snapshot-serengeti} camera survey project across an area of 1.125km\(^2\) in the Tanzania’s Serengeti National Park collected 1.2 million image sets, each containing 1-to-3 images in a space of 3 years [55]. To understand how time-consuming and labour intensive this manual process could be, it took a team of 28000 and about 40000 registered and unregistered users respectively to annotate a 6-month batch of the Serengeti dataset [27]. This observation justifies the need to automate the process of image annotation, and species identification and monitoring.

The most widely used automation technique has been deep learning since 2012 when it broke accuracy records in ImageNet classification challenge [33] and speech recognition [22]. Deep Learning has continued to register tremendous success in several fields including ecology. For example, Barré et al. used field-photographed leaves to develop an automatic plant species identification deep learning model which registered an average classification accuracy of 97.8\% in the top-5 [6]. Several studies have also used animal sounds to build models for identifying and monitoring different species [12, 32, 39]. Deep learning techniques have also been applied in identifying and counting several species in camera-trap images [60, 47, 9]. A study by Ditria et al. compared the identification speed and accuracy of deep learning methods against marine experts and citizen scientists in determining fish abundance in image and video underwater captured data and found that the deep learning algorithm performance was 7.1\% and 13.4\% better than experts and citizen scientists respectively for the image dataset, and 1.5\% and
7.8% better for the video dataset [11]. As with other wild species, monitoring birds should also be a regular ecological activity. In this paper, we propose to study and evaluate state-of-the-art deep learning architectures to detect birds in the webcam captured images. The main focus is to compare how well these architectures can detect birds in images captured from different environments plus their speed and memory consumption. Apart from ecological research and avian protection, bird detection is also important in other multiple applications such as wind energy firms where detection systems are needed to prevent the collision of birds with wind turbines and aviation safety. In aviation, machine learning-based systems are used in differentiating radar signals of birds from abiotic objects [43, 10].

Although numerous studies have been carried out on the application of deep learning techniques to automate the process of bird detection and counting [53, 41, 2, 7, 58], the studies have mainly focused on the use of satellite captured, camera-trap, or unmanned aerial vehicle images. This study uses webcam based images. And thus we propose a first attempt at contributing towards real-time monitoring as opposed to images collected in the past (e.g. camera traps). To the best of our knowledge this is the first study to use webcam-based images. Our study uses the same methods as [23] but with different dataset size and the nature of images used. Hong et al. used fewer and camera trap images [23]. Our proposed approach also provides a broader comparison between several deep learning architectures unlike Hong et al. who studied only three architectures, ResNet-101, Inception V2, and MobileNet v2 [23]. In this study, we used Faster R-CNN and SSD meta-architectures in combination with MobileNet, ResNet50, ResNet101, ResNet152, and Inception ResNet feature extraction networks. Through this study, we also managed to collect and manually verify 10592 images which was contributed to an open source platform for other researchers to use.

The rest of the paper is planned as follows. Section 2 summarizes the main concepts of this study and provides a theoretical background on machine learning, artificial neural networks, and deep learning with a main focus on convolutional neural networks. The section further discusses the state-of-the-art object detection meta-architectures, feature extraction networks, and evaluation metrics. In Section 3, we introduce the dataset, pre-processing techniques, and TensorFlow object detection API used throughout this study. It also presents the implementation details of the models used. Section 4 presents a detailed analysis of the models’ performance and a comparison among each other. Lastly, Section 5 summarizes the paper.

2 Material and Method

2.1 Deep Learning

Deep learning [36] is a subset of machine learning that involves the training of multi-layered artificial neural networks. A neural network with at least two hidden layers is referred to as a deep neural network [46]. In recent years, deep learning algorithms have become more attractive to researchers compared to the conventional machine learning algorithms [36], and this is due to;

1. Increase in the computation power in terms of the graphical processing unit (GPU) and central processing unit (CPU).

2. Availability of huge, well-maintained, and public datasets like Microsoft’s common object and context (COCO) dataset [40], ImageNet [28], MNIST handwritten digit database [37] and many others.
3. The development of novel state-of-the-art algorithms such as AlexNet [33], residual networks (ResNets) [19], GoogLeNet [8], and region-based CNN [51, 52, 19] which not only outperformed the conventional machine learning algorithms but also surpassed humans in the field of classification and recognition [54, 59, 61].

2.2 Convolutional neural networks

A convolutional neural network (CNN) is a deep learning algorithm that has registered state-of-the-art results on real-world applications such as image classification [19], object detection [49], semantic segmentation [42] and speech recognition [1]. Convolutional neural networks were first introduced in the Kunihiko and Sei neocognitron [35] and later modified by Sackinger et al. [38] to a LeNet-5 architecture which registered tremendous success in recognizing handwritings. The application of CNN has been popularized in the computer vision community since the 2012 ImageNet challenge when AlexNet registered outstanding results [33]. In problems like image classification, CNNs algorithms have even outperformed humans [19]. A typical architecture of a CNN is divided into two parts, feature extraction layers and the classifier. The feature extraction part is mainly made of convolutional and pooling layers while the feature mapping is in most cases composed of fully connected layers.

2.3 Meta-architectures

2.3.1 Faster R-CNN

Faster R-CNN [49] is a two-stage CNN meta-architecture composed of a Region Proposal Network (RPN) and the Fast R-CNN detector network. The first stages known as the region proposal network (RPN) uses feature maps generated by feature extraction networks (discussed below) to produce regions of interest (RoI) through a series of fully connected and max-pooling layers [49]. RoIs are proposed candidate object regions that are thought to contain the object being investigated. RPN produces many proposals with potentially a large number of overlapping areas and these multiple detections per image are removed using a non-maximum suppression (NMS) technique [24]. Finally, the proposed regions are fed into a second stage, called Fast R-CNN detector, which predicts whether a bird is contained in the RoI or not. The RPN and Fast R-CNN detector are merged into a single network through sharing their convolutional features [45]. The combination of the two helps Faster R-CNN to achieve better accuracy than the single-stage networks but the accuracy comes at the expense of speed [24].

2.3.2 Single Shot Detector

The Single Shot Detector (SSD) [62] is a single-stage object detection model based on a feed-forward convolutional network that predicts the presence of an object(s) independently in images using multi-scale convolutional bounding box outputs (multi-scale feature maps). An input image and its ground truth boxes are passed through multiple convolutional layers of the backbone network extracting feature maps at different points. Each location of these feature maps is evaluated using different scale filters although the 4 × 4 and 8 × 8 filters are used most often [62, 34] to judge a small set of the default boxes (equivalent to anchor boxes of the Faster R-CNN). The default boxes are attentively selected bounding boxes based on their positions, sizes, and aspect sizes across the targeted image [62]. For every default box, both bounding box offsets and the confidences (or the class probabilities) are predicted. The final detection is decided by the non-maximum suppression algorithm. The SSD network has been used in
several object detection studies and it has produced highly competitive results [23, 29, 31]. Wei et al. compared the performance of SSD against its object detector counterparts in terms of accuracy and speed and found that it was favorably competitive [62].

2.4 Feature extractors

2.4.1 Residual networks

Residual networks (ResNets) were first presented by He et al. in 2015 and at the time the authors had reported improved results on the ImageNet dataset [19]. They presented a 152 layer network that was 8 times deeper than the VGGNets. This network achieved a Top-5 error of 3.5% and this result won the 2015 ILSRVC classification challenge. The Top-5 error is the percentage of the time that the classifier did not include the correct class among its top 5 guesses [33]. Submissions based on this deep ResNets architecture went on to win several other challenges including: the COCO detection and segmentation, and ImageNet object detection and localization challenges. There are three types of residual networks, namely: ResNet50, ResNet101 and ResNet152. In this study, we are going to investigate the performance of all three residual networks as presented by He et al. [20]

2.4.2 MobileNet

MobileNet [25] is a lightweight feature extraction network designed for use in limited memory systems. The model is based on the depth-wise separable convolutions\(^3\) [25] and it factorizes a standard convolution into a depth-wise convolution and a \(1 \times 1\) point-wise convolution (Conv). All layers of the MobileNet are followed by batch normalization (BN) and ReLU activation function apart from the fully connected layer. It is mainly used to design machine learning mobile applications and it was the first TensorFlow computer vision model. It also reduces the computational cost and number of parameters drastically compared to ResNets and VGGNets but with same number of input and output channels [21].

2.4.3 Inception ResNet

At the ILSVRC competitions of 2014, Christian et al. presented a high performance deep CNN architecture named “Inception” that demonstrated improved computational cost compared to ResNets [8]. The original network used three different convolutions \(1 \times 1\), \(3 \times 3\), and \(5 \times 5\). In 2015, Szegedy et al. proposed several changes to the inception architecture to reduce computational complexity and improve the computational speed, and accuracy [56]. The changes included: replacing the \(5 \times 5\) convolution with two \(3 \times 3\) convolutions and factorizing \(n \times n\) convolutions to \(1 \times n\) and \(n \times 1\) combination. Szegedy et al. [56] found out that their method was six times cheaper computationally and used at least five times less parameters than the best ResNet of He et al. [19].

2.5 Transfer Learning

When training deep learning networks to solve specific problems one of the two problems may arise. The first is not having enough labeled data and the second is having to train a deep network from scratch. Not having enough data would force us to collect and annotate large

\(^3\)The depth-wise separable convolution gets its name from the fact that, it splits a kernel into 2 separate kernels namely: the depth-wise convolution and the point-wise convolution [21]
amounts of data but in most cases the data may not be available. Training a deep network from scratch is a challenging problem because the process may take hours or even days since these weights begin with random values. The optimizer takes a lot of time to converge if at all these initialized values are far from the optimal solution [13]. One of the ways used to overcome both problems is by utilizing the network weights from pre-trained models. This process is also known as transfer learning. Transfer learning is a deep learning technique that allows leveraging the knowledge generated from previous training to a new but related problem [4]. We make an assumption that many of the factors that explain the variations in the earlier problem are relatively similar to variations that need to be captured for learning the new problem. In this study, we used transfer learning by repurposing weights pre-trained on MS COCO dataset [40] to our novel dataset. MS COCO has been used as a benchmark dataset for many object detection researchers because it has proportionately more instances per category than any other available public datasets like PASCAL VOC [14] and also contain more objects (7.7 per image) than the popular ImageNet and PASCAL VOC with 3 and 2.3 objects per image respectively [40].

2.6 Evaluation protocol

In this study, all the models were evaluated using the performance evaluation tool as the MS COCO object detection challenge [40]. The main evaluation metric of the tool is mean average precision (mAP), which is averaged using 10 IoU thresholds, i.e. \( \text{IoU} = \{0.50, 0.55, \ldots, 0.95\} \) in increments of 0.5. Additionally, this MS COCO performance measure also evaluates average precision (AP) and average recall (AR) depending on object bounding box sizes like small (area \(< 32^2\)), medium (\(32^2 < \text{area} < 96^2\)), and large (area \( > 96^2\)), and varying AR detection per images, i.e. AR\(_1\), AR\(_{10}\), AR\(_{100}\) representing AR given 1, 10, and 100 objects detections per image respectively.

3 Experimental Framework

3.1 Dataset

We trained all the detection models on images collected from the live feed watcher cams (https://www.allaboutbirds.org/cams/) of Cornell Lab of Ornithology situated in 6 unique locations around the world. The Cornell Lab of Ornithology is an institute dedicated to biodiversity conversation with the main focus on birds through research, citizen science, and education. In total, 10592 images were collected for this study and Figure 1 shows some of the collected images. After using the dataset, it was contributed to an open source platform called Zenodo where those who wish to use it for research purposes are free to do so and it can be accessed at this link: https://zenodo.org/record/5172214#.YVTaQZpBxhH.

3.2 Hardware

We ran the experiments on an MSI GL75 Leopard 10SFR laptop with CUDA 11.0, cuDNN SDK 8.0.4, and Windows 10 x64. The hardware configuration of the laptop is as follows: 10th Gen Intel Core i7-10750H, GeForce RTX 2070 8GB GDDR6 GPU card, and 32GB DDR4 RAM.
3.3 Experiments Details

Based on the studies discussed in introduction, we selected two meta-architectures, namely, Faster R-CNN and SSD implemented using the following feature extractors, MobileNet, ResNet50, ResNet101, ResNet152, and Inception ResNet v2. The choice of the feature extractors was based on the reported outstanding results in a number of studies [49, 62, 26, 19, 8]. Through transfer learning, all the feature extraction networks were initialized with weights pre-trained on the MS COCO dataset provided by the TensorFlow object detection API.

3.3.1 SSD Models

To achieve the best detection results on our dataset, during training and hyper-parameter tuning of the models, we followed the experimental procedural setup as used by He et al. [20] and Huang et al. [26] because the good performance achieved. The hyper-parameters of the networks were configured as shown in Table 1. The models were fine-tuned until satisfactory results (i.e. accuracy and speed) were obtained.

3.3.2 Faster R-CNN

During training and hyper-parameters fine-tuning of all the four Faster R-CNN, we followed closely the configuration procedures of Huang et al. [26], Ren et al. [49], and He et al. [20]. The best models had hyper-parameters set as shown in Table 2. For the training time, Figure 2

---

4 https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_detection_zoo.md
Automating Bird Detection using Deep Learning

<table>
<thead>
<tr>
<th>Parameters</th>
<th>MobileNet</th>
<th>ResNet50</th>
<th>ResNet101</th>
<th>ResNet152</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image size</td>
<td>320 × 320</td>
<td>320 × 320</td>
<td>320 × 320</td>
<td>320 × 320</td>
</tr>
<tr>
<td>Epochs</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Kernel size</td>
<td>3 × 3</td>
<td>3 × 3</td>
<td>3 × 3</td>
<td>3 × 3</td>
</tr>
<tr>
<td>Optimizer</td>
<td>SGD</td>
<td>SGD</td>
<td>SGD</td>
<td>SGD</td>
</tr>
<tr>
<td>Momentum</td>
<td>0.9</td>
<td>0.9</td>
<td>0.9</td>
<td>0.9</td>
</tr>
<tr>
<td>Batch size</td>
<td>16</td>
<td>16</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>Learning rate</td>
<td>0.08</td>
<td>0.04</td>
<td>0.08</td>
<td>0.003</td>
</tr>
<tr>
<td>Iterations</td>
<td>50,000</td>
<td>30,000</td>
<td>27,000</td>
<td>120,000</td>
</tr>
<tr>
<td>Dropout probability</td>
<td>0.8</td>
<td>0.8</td>
<td>0.8</td>
<td>0.8</td>
</tr>
<tr>
<td>Weight decay</td>
<td>0.0004</td>
<td>0.0004</td>
<td>0.0004</td>
<td>0.004</td>
</tr>
</tbody>
</table>

**Post-processing**

| Intersection over union | 0.6 | 0.6 | 0.6 | 0.6 |
| Score threshold         | $1 \times 10^{-8}$ | $1 \times 10^{-8}$ | $1 \times 10^{-8}$ | $1 \times 10^{-8}$ |

Table 1: The hyper-parameter tuning values for both training and post-processing of the models trained using SSD meta-architecture in combination with MobileNet, ResNet-50, ResNet-101, and ResNet-152 feature extractors. The batch size for ResNet-101 and ResNet-152, was reduced to 8 due to memory constraints.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>ResNet50</th>
<th>ResNet101</th>
<th>ResNet152</th>
<th>ResNet-V2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image size</td>
<td>1024 × 1024</td>
<td>1024 × 1024</td>
<td>640 × 640</td>
<td>320 × 320</td>
</tr>
<tr>
<td>Epochs</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>10</td>
</tr>
<tr>
<td>Stride</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Kernel size</td>
<td>2 × 2</td>
<td>2 × 2</td>
<td>2 × 2</td>
<td>2 × 2</td>
</tr>
<tr>
<td>Anchor size</td>
<td>16 × 16</td>
<td>16 × 16</td>
<td>16 × 16</td>
<td>3 × 3</td>
</tr>
<tr>
<td>Optimizer</td>
<td>SGD</td>
<td>SGD</td>
<td>SGD</td>
<td>SGD</td>
</tr>
<tr>
<td>Momentum</td>
<td>0.9</td>
<td>0.9</td>
<td>0.9</td>
<td>0.9</td>
</tr>
<tr>
<td>Batch size</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Learning rate</td>
<td>0.004</td>
<td>[0.002, 0.0002]</td>
<td>[0.004, 0.0004]</td>
<td>[0.03, 0.003]</td>
</tr>
<tr>
<td>Iterations</td>
<td>20,000</td>
<td>[10,000 &amp; 7,000]</td>
<td>[20,000 &amp; 10,000]</td>
<td>[40,000 &amp; 15,000]</td>
</tr>
</tbody>
</table>

**Post-processing**

| Intersection over union | 0.7 | 0.7 | 0.7 | 0.7 |

Table 2: The hyper-parameter tuning values for both training and post-processing of the Faster R-CNN models. Initially, the ResNet-101 model was trained for 10,000 iterations at a learning rate of 0.002, and then decreased to 0.0002 for the next 7,000 iterations. For the ResNet152 model, it trained initially for 20,000 iterations at a learning rate of 0.004 and 0.0004 for the next 10,000 iterations. Due to memory constraints, the image size for the Inception ResNet-V2 model was reduced and the model trained for 40,000 and 15,000 iterations at a learning rate of 0.03 and 0.003 respectively.

shows that Faster R-CNN models trained faster than the SSD models except for the SSD with MobileNet which took the shortest training time of four hours. Looking at the figure, larger feature extractors took more time to train than the smaller ones.
In this section we give comprehensive discussion and comparison of results achieved by all the models. The overall results show that Faster R-CNN models achieved better results (in terms of detection accuracy) ranging from 75.2% to 92.3% than the SSD models whose range of detection accuracy was between 67.5% and 89.4%. The performance of the Faster R-CNN models in detecting both large and small birds was better than that of the SSD models as shown in Table 3. The Faster R-CNN model trained with the ResNet152 feature extractor yielded better results across all the evaluation metrics. It achieved an overall mean average precision of 92.3% but still, the other models also registered good performances. It is also clear that across all models the highest average precision was obtained when the IoU was set to 0.5 (AP$_{50}$) as opposed to 0.75 IoU.

In terms of speed, SSD models were remarkably faster than the Faster R-CNN models as
shown in Table 4. It is also observed in Table 4 that the deeper the backbone network, the slower it gets to train. For example, the ResNet-50 model is faster to train than ResNet-152 in both SSD and Faster R-CNN meta-architectures. This is because deeper networks need more parameters to learn [5]. It can also be seen that the SSD model with MobileNet feature extractor is the smallest in terms of inference graph file size compared to all other models. Therefore, this model can be deployed in memory-constrained systems because its light, fast (110ms) and registered quite a fair mAP of 67.5%. These results also show that SSD models consume more memory than the Faster R-CNN with the same feature extraction networks e.g. SSD combined with ResNet-152 consumes 37MB more than Faster R-CNN with the same base network.

Table 4: Inference time per image on each meta-architecture and the models’ parameter file sizes.

<table>
<thead>
<tr>
<th>Meta-architecture</th>
<th>Extractor</th>
<th>Model size (MB)</th>
<th>Inference time (ms/image)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSD</td>
<td>MobileNet</td>
<td>30.5</td>
<td>110</td>
</tr>
<tr>
<td></td>
<td>ResNet50</td>
<td>258</td>
<td>165</td>
</tr>
<tr>
<td></td>
<td>ResNet101</td>
<td>405</td>
<td>183</td>
</tr>
<tr>
<td></td>
<td>ResNet152</td>
<td>532</td>
<td>197</td>
</tr>
<tr>
<td>ResNet50</td>
<td>MobileNet</td>
<td>221</td>
<td>251</td>
</tr>
<tr>
<td></td>
<td>ResNet101</td>
<td>371</td>
<td>270</td>
</tr>
<tr>
<td></td>
<td>ResNet152</td>
<td>495</td>
<td>283</td>
</tr>
<tr>
<td></td>
<td>Inception ResNet-V2</td>
<td>469</td>
<td>256</td>
</tr>
</tbody>
</table>

Figure 3: Difference in speed and accuracy between the object detection models.
Figure 4: **Left:** Faster R-CNN predictions, **right:** SSD predictions. The top two images contain 2 birds (Faster R-CNN detected both, SSD detected one), and the bottom two contain 5 birds (Faster R-CNN detected all, SSD detected 3).

Figure 3 demonstrates the trade-off in speed and accuracy. It is seen that Faster R-CNN models, in particular Faster R-CNN with ResNet101 and ResNet152, have the highest accuracy but their accuracy came at the expense of speed as they exhibited the worst testing time. In terms of speed, the SSD models proved to be faster than the Faster R-CNN models.

The Faster R-CNN models also outperformed the SSD models in detecting small and overlapped birds in images. In the Figure 4, we show some of the examples where the Faster R-CNN model did well in detecting the small and overlapped birds compared to the SSD.

The Faster R-CNN model with the ResNet152 feature extractor- our best performing model, in terms of detection accuracy- was subjected to different images randomly obtained from Google with environmental conditions not captured in the test set images. This was done to determine how well the model would respond to these new domains. The model’s performance was extremely good in detecting birds especially those taken at night/with dim light, images of birds with people, and images of birds captured at different angles. Therefore, this indicates that our model can be used to detect birds in different environment settings.

### 5 Conclusion

This study focused on studying, designing, and evaluating state-of-the-art deep learning object detection algorithms that are capable of detecting birds in webcam-captured images. We conclude that the Faster R-CNN combined with the ResNet152 feature extractor as the best for achieving the highest mean average precision compared to other architectures, and the SSD
with MobileNet as the best model in terms of speed and smaller memory consumption. The state-of-the-art results obtained in this study confirm that deep learning-based algorithms are capable of detecting birds of different sizes in different environments and we recommend that our best overall model can be used by ecologists in monitoring and identifying birds from other species.

References


[17] BJ Godley, JM Blumenthal, AC Broderick, MS Coyne, MH Godfrey, LA Hawkes, and MJ Witt. Satellite tracking of sea turtles: where have we been and where do we go next? Endangered species


[34] Ashwani Kumar, Zuopeng Justin Zhang, and Hongbo Lyu. Object detection in real time based on improved single shot multi-box detector algorithm. *EURASIP Journal on Wireless Commu-
Automating Bird Detection using Deep Learning


[52] Girshick Ross and Donahue Jitendra. Region-based convolutional networks for accurate object detection and segmentation. IEEE transactions on pattern analysis and machine intelligence,


