Smart Camera Design for Real-Time Bird and Cat Detection

# Project Brief

Bob the bird keeper has a problem that he thinks you can help him with. His birds are constantly harassed by the neighbor’s cats who are looking for a quick dinner. To protect his birds, Bob has setup a camera triggered alarm that goes off when a cat is nearby. In order to help Bob, please design and implement an automated algorithm that triggers when a cat is in view of the camera. To reassure him that his birds are alive and well, please also detect if a bird is present in the frame of view.

# Aim

The project brief consists of developing smart software for Bob’s camera to track his birds and detect the neighbor’s cats. To help Bob, the engineer chose to design a computer vision (CV) application for his camera to inform him of the cat’s presence and watch his bird’s while he is gone.

# Introduction

CV consists of building mathematical models to recognize image patterns (Szeliski, 2011) such as the shape of objects, lighting and colour distribution. The inputs for CV include two-dimensional (2D), three-dimensional (3D) images (stereoscopic vision) position (Sabbatelli, 2015) and optic flow (OF) (Davies, 2012). The mathematical description of an image is a 2D map of intensities and colours (Klette, 2014) on which operations can be performed to retain specific features of the latter. Those operations include as convolution, correlation, filtering in the frequency domain (Jain, 1989; Szeliski, 2011), pyramidal expansion and reduction (Burt & Adelson, 1983) as well as wavelet and Fourier transforms.

In the past ten years, computers became powerful enough to run and train convolutional neural networks (CNN), which can identify patterns in images (Kuo, 2016). The basic CNN architecture has one convolutional layer, one pooling layer and one fully connected layer, which feeds into an artificial neural network (ANN) (O’Shea & Nash, 2015). The CNN stacks “learnable kernels”, which are image fragments the CNN learns to recognize in a 2D activation map (O’Shea & Nash, 2015). These kernels form the artificial neurons of the CNN and apply “REctified COrrelations on a Sphere” (RECOS) algorithm, which assesses how similar a detected feature is to the node’s assigned kernel (Kuo, 2016). Then these features are highlighted or filtered out depending on their importance by the CNN’s weights optimized with backpropagation (Gurney, 1997). Then the pooling layer flattens the kernels into a one-dimensional (1D) array constituting the input layer an ANN connected to the convolutional architecture, which feeds the latter’s hidden layers and output layers (Tarassenko, 1998). Multiple convolutional, pooling and full connection layers can be added to improve the CNNs performance (Albelwi & Mahmood, 2017).

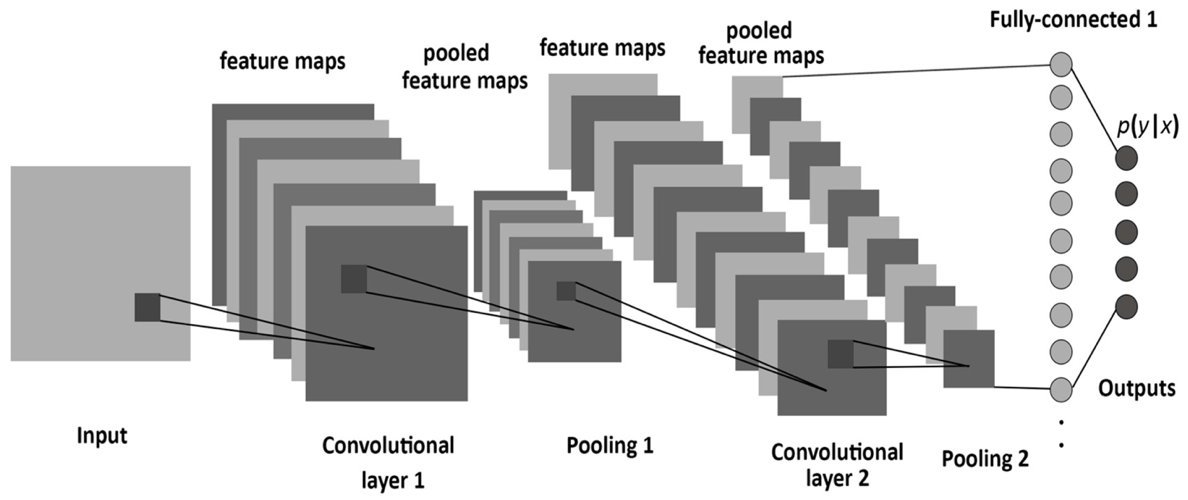
**

Figure 1 CNN with two convolutional layers and two pooling layers connected to an ANN (Albelwi & Mahmood, 2017).

In this project the camera’s input will be an OF or a flux of images (Szeliski, 2011) as Bob must be alerted when the neighbour’s cat harasses his birds in real time. Thus, the smart camera application must include two components, a CNN to recognize the cats and the birds and an outline detector to recognize moving objects.

There are two state of the art techniques for moving object recognition, the first is single shot detection (SSD) (Liu, et al., 2016) and the second is the You Only Look Once (YOLO) method. In this project, only the YOLO method is discussed as SSD was not implemented due to the GPU requirement for training.

YOLO is a regression-based object recognition method like SSD, which simultaneously detects moving object and identifies them. Those methods are opposed to classification-based methods, which is the method implemented in this project to compare my design to a state-of-the-art solution. YOLO draws a bounding box around an object by spatially separating bounding boxes with the likelihood they belong to a specific class (Handalage & Kuganandamurthy, 2021).

# Methods

Two smart camera softwares were developed, one using the YOLO algorithm to simultaneously recognize the location of a bird or a cat, and the other combining a CNN classifier with image operations to recognize the latter.

## YOLO

This implementation requires a functional computer webcam, OpenCV and the imageai software packages to run. A pre-trained open-source YOLO CNN, which can identify 80 different objects including birds and cats, was designed for the imageai package by Moses Olafenwa (Olafenwa & Olafenwa, 2021). Thus, a GPU is not needed to implement YOLO and it straightforward to pair this CNN with OpenCV and a PC Webcam.

## Image Operations and CNN

This implementation divides the bird and cat recognition problem into to two parts and requires access to the OpenCV, Tensorflow and Keras software packages. The first part consists of making outlines for picture as the engineer does not have a cat or bird to test his software on and he will print pictures of cats and birds on piece of paper to test the application built. The second step is to train and design a CNN capable of distinguishing between birds and cats for this purpose the engineer was handed down a dataset folder containing a training and a test set containing pictures of birds and cats by his supervisor to train the CNN.

The pictures can be detected by using simple image operations to detect rectangles with image filters implemented in the python-opencv package. In this case, first the images were converted into grayscale, then a bilateral filter increased the edge contrasts, a gaussian blurred image contents and a Canny filter to preserve only the contours. After performing those operations, it then becomes possible to use openCV’s shape detector for rectangles to extract image outlines and reshape them to match the CNN input size, which is a 64 by 64 by 3 image.

The CNN architecture consisted of one convolutional layer with 64 rectifier-linear activation unit (Relu) nodes one for each pixel followed by a pooling layer breaking down the image into 32 Relu kernels. Those kernels are fed to a second convolutional layer with 32 Relu units connected to a Relu pooling layer and a flattening layer. This layer is connected to a hidden layer with 140 neurons connected to two sigmoidal output nodes. The CNN was then trained for 40 epochs with 680 steps for validation and training.

The engineer also attempted to use the same CNN architecture with on sigmoidal output node with the same training parameters to experiment. However, this CNN had a major design flaw for this task as anything that was not detected as a bird would be a cat. This design would bring poor for Bob’s user experience as it may constantly indicate that there is a cat.

# Results

To test both networks after training, the “Image Recognition Test Sheet” available in appendix is shown to the webcam to test the application’s functionality.

## YOLO

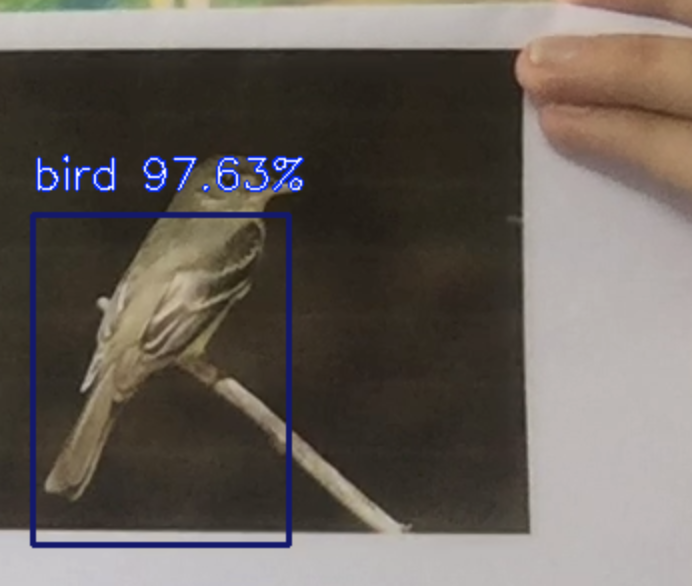


Figure 2 YOLO Sees a Bird with 97.63% Certainty



Figure 3 YOLO Sees a Bird with 89.79% Certainty

## Image Operation and CNN

The one sigmoid node output has an accuracy of 97.9% after training.

The two sigmoid node output has an accuracy of 98% after training.



Figure 4 Two Node Output CNN Sees a Bird

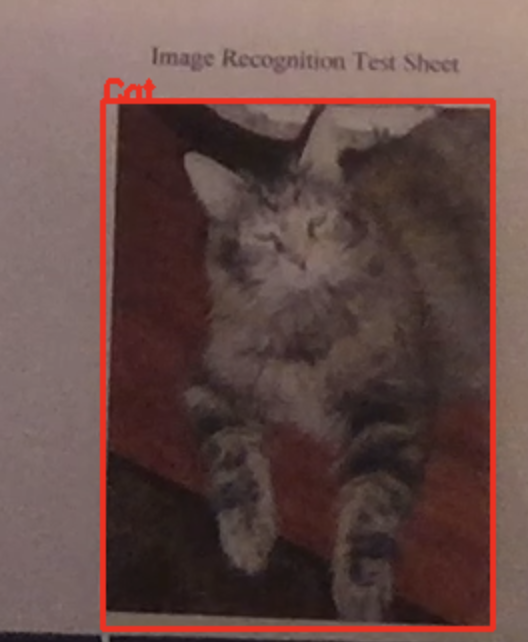


Figure 5 Two Node Output CNN Sees a Cat

# Discussion

Both Networks are able to recognize the cat and the bird in real-time. However, for this application the CNN the engineer trained is better as the webcam application is able to run more smoothly than with the YOLO CNN. This guarantees Bob will be able to watch his birds and keep an eye on the cat while not being present. More hyperparameter tuning can also be done to improve outline detection and the algorithm’s unfriendliness to zoom and lighting, which can cause bad predictions. Thus, when testing the software, the image should remain at a stable distance from the webcam to ensure things run smoothly.

# References

1. Albelwi, S. & Mahmood, A., 2017. A Framework for Designing the Architectures of Deep Convolutional Neural Networks. *Entropy,* 24 May, 19(6), pp. 1-10.
2. Burt, P. J. & Adelson, E. H., 1983. The Laplacian Pyramid as a Compact Image Code. *IEEE Trans. Communications,* 9(4), pp. 532-540.
3. Davies, E. R., 2012. *Computer & Machine Vision: Theory, Algorithms, Practicalities.* 4th Edition ed. London: Elsevier.
4. Gurney, K., 1997. *An Introduction to Neural Networks.* s.l.:CRC Press Taylor&Francis Group.
5. Handalage , U. & Kuganandamurthy, L., 2021. *Real-Time Object Detection using YOLO: A review,* Malabe: Research-Gate.
6. Jain, A. K., 1989. *Fundamental of Digital Image Processing.* 1st Edition ed. Upper Saddle River: Prentice-Hall Inc..
7. Klette, R., 2014. *Concise Computer Vision.* 1st Edition ed. London: Springer.
8. Kuo, C.-C. J., 2016. Understanding Convolutional Neural Networks with A Mathematical Model. *Journal of Visual Communication and Image Representation,* November, 41(1), pp. 406-413.
9. Liu, W. et al., 2016. *SSD: Single Shot MultiBox Detector.* [Online]   
   Available at: https://arxiv.org/abs/1512.02325  
   [Accessed 26 May 2021].
10. O’Shea, K. & Nash, R., 2015. *An Introduction to Convolutional Neural Networks,* New-York: Cornell University Library.
11. Olafenwa, M. & Olafenwa, J., 2021. *ImageAI.* [Online]   
    Available at: https://imageai.readthedocs.io/en/latest/video/index.html  
    [Accessed 16 May 2021].
12. Sabbatelli, M., 2015. *3D Vision-based Perception and Modelling techniques for Intelligent Ground Vehicles.* [Online]   
    Available at: https://www.researchgate.net/figure/Pinhole-camera-model\_fig2\_280546809  
    [Accessed 8 March 2018].
13. Szeliski, R., 2011. *Computer Vision: Algorithms and Applications.* 1st Edition ed. Washington: Springer.
14. Tarassenko, L., 1998. *A Guide to Neural Computing Applications.* 1st Edition ed. London: Elsevier Inc..

# Appendix: Image Recognition Test Sheet



