A Convolutional Neural Network for Automatic Analysis of Aerial Imagery

Frederic Maire, Luis Mejias
School of Electrical Engineering and Computer Science, Queensland University of Technology, Brisbane, Australia
f.maire@qut.edu.au, luis.mejias@qut.edu.au

Amanda Hodgson
Murdoch University Cetacean Research Unit, Murdoch University, Perth, Australia
a.hodgson@murdoch.edu.au

Abstract—This paper introduces a new method to automate the detection of marine species in aerial imagery using a Machine Learning approach. Our proposed system has at its core, a convolutional neural network. We compare this trainable classifier to a handcrafted classifier based on color features, entropy and shape analysis. Experiments demonstrate that the convolutional neural network outperforms the handcrafted solution. We also introduce a negative training example-selection method for situations where the original training set consists of a collection of labeled images in which the objects of interest (positive examples) have been marked by a bounding box. We show that picking random rectangles from the background is not necessarily the best way to generate useful negative examples with respect to learning.

I. INTRODUCTION

This paper introduces a technique that can be used to automate the processing of images analysis taken during an aerial survey using a custom payload onboard an Unmanned Aerial Vehicle (UAV). The amount of data produced by such flights is considerable (tens of thousands of images) making the process of manual review intractable [1]. Similar applications of UAVs for surveillance and monitoring in areas such as agriculture, law enforcement, equipment and infrastructure inspection, etc., could benefit from automated image analysis. This type of automation could radically reduce the human hours needed to perform tasks in these fields. Our particular interest was in the automatic detection of marine mammals in images taken from an aircraft (manned or unmanned). We introduce a machine learning approach using convolutional neural networks that automatically detect and annotate marine mammals (dugongs) in images. In the area of vision-based marine mammal identification, most image processing techniques investigated to date fall in the category of low-level image processing [2], [3], [4], [5], [6]. Whilst straightforward, this approach is prone to high rates of false detection which makes the approach in some instances unreliable. Machine learning, and in particular Convolutional Neural Networks (CNNs) has the potential to provide enormous improvements in the automated detection of marine fauna, and other similar applications.

A. The specific challenges of marine mammal detection

What makes the detection of dugongs particularly challenging is that their appearance varies dramatically with the sea conditions. Their apparent color changes with the depth and the turbidity of the water. Although the shape of a dugong is relatively rigid, their tail is not always visible. Moreover, parts of their bodies can be covered by small waves with breaking crests or whitecaps as can be seen in Figure 1. The appearance of the dugongs depends also on the sea floor. In Figure 2, some of the grazing dugongs are hardly distinguishable from the background.

This paper is structured as follows. Section II reviews recent work in CNN for image analysis. Section III describes the CNN approach proposed this paper. Section IV describes how additional training samples are generated. Section V outlines the negative example-selection method. Section VI presents the outcomes and analysis of data. Finally, section VII presents concluding remarks.

II. RELATED WORK

The literature on marine mammal detection using electro optical sensors is not extensive. Infrared and standard cameras have been used to perform detection from aerial platforms [7], [8], [9], [10], [11], [3], [4]. The main limitations of this approach have been identified the environmental conditions, and their effects on illumination within the images. Despite these challenges, visual imagery or vision is an attractive solution given it offers a rich and permanent source of information, and is easily generalisable to many types of aircraft. The analytical approaches presented in [3], [4] uses color segmentation and blob shape analysis. These two attempts represent a significant
step forward in terms of detection rates, however they are still prone to many false positive detections and depend on user tuneable parameters. An alternative to handcrafting image processing operators or features is to learn them.

CNNs have seen many uses in pattern recognition. Face and character recognition are among the most widespread applications of CNNs [12], [13], [14], [15]. Understanding human motion is the focus of Ji et al. [16]. By incorporating spatial and temporal information they develop a 3D CNN to recognise human motion/activity in videos with possible applications in airport security. Detection of pedestrians is another area where CNNs have been tested with excellent results [17]. A variant of a CNN that is accelerated by Graphical Processing Units (GPUs) is proposed by Ciresan et al. [18]. The motivation for this approach is that computational speed is still a limiting factor for CNN architectures. Despite successful uses of CNNs for face and character recognition there has been little use of this technique for surveys of fauna. Our research contributes an automated algorithm for marine mammal detection. We also introduce a negative training example-selection method for situations where the original training set consists of a collection of labeled images in which the objects of interest (positive examples) have been marked by a bounding box.

III. CONVOLUTIONAL NEURAL NETWORKS

By design, CNNs present a higher degree of invariance to small distortions like translation, scaling and skewing than Multi-Layer Perceptrons. Hubel and Wiesel’s pioneering work on the cat’s visual cortex [19] gave a biologically plausible model for CNN. Their key biological observations from a computer science point of view are that:

- One type of biological neuron has a small receptive field and responds to edge-like input patterns.
- Another type has a larger receptive field and is not sensitive to the exact location of the input pattern.

These two types of neurons are present in the popular CNN architecture that was introduced in LeNet [20]. Other related models include the NeoCognitron [21] and HMAX [22].

A. High Level Architecture

The high level architecture of the CNN that we implemented is illustrated in Figure 4. It is a simplified stack of layers based on LeNet [20]. The combination of the first three layers can be interpreted as a feature extractor, whereas the logistic regression layer can be viewed as a simple classifier whose inputs are learned features.
There are three dugongs in this 884 × 580 image. In these calm sea conditions, the dugongs are easier to detect. Notice that the tail of the dugong in the purple rectangle is hardly visible.

The convolutional neural network starts with two LeNet convolutional layers followed by a hidden layer, and finally a logistic regression layer whose output predicts whether or not the input image contains a dugong.

B. LeNet Convolutional Layer

The two LeNet convolutional layers differ only in the size of their weight tensors. Their common structure is sketched in Figure 5. In this figure, the large green square represents the 3D input tensor $x$, the large blue square corresponds to the intermediate feature maps $h$ (another 3D tensor), and the orange square denotes the 3D output tensor $a$ of the LeNet convolutional layer. Formally, the $k^{th}$ intermediate feature map is

$$h^k = \tanh(W^k * x) \text{ where } \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

The symbol * denotes the convolution operator, and $W^k$ represents the 3D tensor associated with the $k^{th}$ intermediate feature map. That is, the $k^{th}$ 2D sub-tensor of the 3D tensor $h$ is the matrix $h^k$. The hyperbolic tangent transfer function $\tanh$ is applied element-wise to tensors. The shape of the tensor $W$ is (number of filters, number of input feature maps, filter height, filter width). For the first LeNet convolutional layer, the parameters are as follows:

- Number of filters = 35. This number was found experimentally.
- Number of input feature maps = 3. This is the number of color channels of the input image.
- Filter height = filter width = 5. Experimentally we observed that larger sizes did not improve the classification performance.

The second operation performed in a LeNet convolutional layer is the application of the max-pooling operator. Max-pooling performs non-linear down-sampling by partitioning the input feature maps into a grid of non-overlapping rectangular cells. For each cell, the maxpool operator outputs the maximum value found in the cell. In our implementation, max-pooling is done over cells of size $2 \times 2$. Formally, the $k^{th}$ output feature map of the convolutional layer is

$$a^k = \text{maxpool}(h^k)$$

Max-pooling is beneficial because this step performs some sort of dimension reduction (reducing the input by a factor of 4 in the case where cells are of size $2 \times 2$), and provides
some form of translation invariance. There are 8 possible 1-pixel translations of the input (left, right, top, bottom and four diagonals). For 3 out of these 8 cases, the maximum element of the cell will still be in the same cell after a 1-pixel translation of the input.

Whereas the two LeNet convolutional layers are sparse, the layers following them are fully-connected. The hidden layer and the logistic regression layer that are shown in Figure 4 are the building blocks of traditional multi-layer perceptrons.

C. Hidden Layer

The penultimate layer named Hidden Layer is the first fully-connected layer. It receives as input a vector obtained by flattening the output tensor from the second LeNet convolutional layer. That is, all the entries of the features maps belonging to the output of the second LeNet convolutional layer are concatenated into a 1D vector. A hidden layer performs on its input vector \( x \) the non-linear transformation

\[
y = \tanh(W x + b)
\]

The general purpose of a hidden layer is to increase the expressiveness of the network so that it can capture the non-linear dependencies between the input variables and the predicted variables (here class labels).

D. Logistic Regression

The final layer of our convolutional neural network is a probabilistic, linear classifier called a Logistic Regression layer. Like the hidden layer, this layer is parameterized by a weight matrix \( W \) and a bias vector \( b \). However, the transfer function \( \tanh \) is replaced by a \( \text{softmax} \) function.

Formally, the output vector \( y \) of the logistic regression layer when presented with a vector \( x \) is derived as

\[
y = \text{softmax}(W x + b)
\]

The \( i^{th} \) entry \( y_i \) of the output vector \( y \) estimates the likelihood of the \( i^{th} \) class given the current input. This probability is computed with the following formula

\[
y_i = \frac{e^{W_{i}:x+b_i}}{\sum_{j=1}^{n} e^{W_{j}:x+b_j}}
\]

where \( n \) is the number of classes and \( W_{i} \) represents the \( i^{th} \) row of the matrix \( W \). The expression \( e^{W_{i}:x+b_i} \) can be interpreted as a measurement of how far from the hyperplane of equation \( W_{i}:x+b_i = 0 \) the input \( x \) is. The distance to this \( i^{th} \) hyperplane reflects the probability of class \( i \) for the given input.

E. Training

As the output layer of the CNN is a logistic regression layer, we apply the negative log-likelihood loss as the training error. A stochastic gradient descent with mini-batches of size 500 is used to update the network weights.

Early-stopping prevents over-fitting by monitoring the CNNs performance on a validation set. That is, a set of examples that we never use for gradient descent. The validation examples are drawn from the same distribution that generates the training set and the test set.

The original dataset was partitioned randomly into 70% training set, 15% validation set and 15% test set. When the CNN performance ceases to improve sufficiently on the validation set, or even degrades with further optimization, the training is stopped. The CNN with the lowest validation error is then returned.

F. Pylearn-Theano Implementation

We wrote our own CNN implementation using Pylearn [23], a machine learning framework/library built on top of Theano [24]. Theano is a mathematical expression compiler for Python that translates high level NumPy-like code into machine language for efficient CPU and GPU computation. A nice feature of the Theano framework is the automatic symbolic computation of the gradient of networked functions. That is, backpropagation update rules come for free.

IV. GENERATING ADDITIONAL IMAGES FOR TRAINING PURPOSE

Training any deep neural network requires a large training set. We have access to about 30 high resolution aerial images. The images are 4288 pixels wide and 2848 pixels high. There are a total of 121 dugongs in the images. Each dugong fits in a window of 100 x 100 pixels. To enlarge this dataset we have applied three types of geometric transformations; 12 rotations with an angle multiple of 30 degrees, scaling by factors of 0.9, 1.0 and 1.1, and axial vertical symmetry.

For each original training window, we generate \( 72 = 12 \times 3 \times 2 \) additional windows by applying these geometric transformations. Starting from the set \( W^0_p \) of positive windows (those containing a dugong), we apply to each window in \( W^0_p \) the 12 rotations to derive a super-set \( W^1_p \) of positive windows. Then we apply the 3 scaling transformations to the windows in \( W^1_p \) to derive a super-set \( W^2_p \) of positive windows. And finally, we apply the identity and the axial symmetry to \( W^2_p \) to derive another super-set \( W^3_p \). The added benefits of applying all the rotations to the original input windows is that the classification by the CNN becomes invariant to the rotation of its input. Similary the scaling and reflection operations improve the generalization capability of a CNN trained on a relatively small dataset by deep learning standards.
V. AUTOMATIC SELECTION OF RELEVANT NEGATIVE EXAMPLES

Once a human expert has painstakingly marked a set of images by drawing bounding boxes (windows) around the animals, it is easy to derive a training set of positive and negative examples. Simply, take the sub-images defined by the bounding boxes as positive window examples, and randomly select background windows that do not intersect any of the marking bounding boxes as negative window examples.

When negative training examples (windows) are drawn randomly from the background, it is unlikely that they will be ambiguous. A negative training example is valuable for training purposes if it is not a dugong but vaguely resembles a dugong. These examples are not common. They occur near wave crests and in areas where the seafloor is not uniform. To help build a robust classifier that can discriminate dugongs from their background, it is beneficial to have positive and negative training examples that are difficult to classify. Figure 6 shows some hard to classify windows.

Algorithm 1 outlines a method to automatically retrain a classifier when the training dataset available consists of a collection of images with objects of interest marked with bounding boxes.

Algorithm 2: CNN Based Dugong Detector

VI. EXPERIMENTAL RESULTS

The objective of our study was to explore whether CNN could be competitive compared to handcrafted classification methods for the detection of dugongs in aerial images. The trained CNN classifies 28 by 28 input windows as either containing a dugong or not. Algorithm 2 processes a large image by scanning a test window over the image and feeding the sliding window to a CNN.

We explored different sets of architectural parameters for the CNN. The search was not exhaustive as training a CNN can take a couple of days. The best set of parameters we found was as follows. Both convolutional layers had 35 feature maps. The filter height and width for the convolutional layers were set to 5. The number of neurons in the hidden layer was 110.

input : \( I_{rgb} \): a RGB color image, \( \mathbb{N} \): a convolutional neural network, \( \theta_p \): a probability threshold, \( \theta_a \): an area threshold

output: \( L \): a list of blob bounding boxes believed to contain dugongs

1. begin
2. \( I_p \leftarrow \) zero matrix (same size as \( I_{rgb} \))
3. \( s_y \leftarrow \) half of the horizontal size of the \( \mathbb{N} \) input
4. \( s_x \leftarrow \) half of the vertical size of the \( \mathbb{N} \) input
5. for \( y \) in \( Y_{range} \)
6. \( \quad \) for \( x \) in \( X_{range} \)
7. \( \quad \quad \quad I_{patch} \leftarrow I_{rgb}(y-s_y \ldots y+s_y, x-s_x \ldots x+s_x) \)
8. \( \quad \quad \quad I_p(y, x) \leftarrow \) likelihood \( I_{patch} \) contains a dugong according to \( \mathbb{N} \)
9. end
10. end
11. \( I_b \leftarrow \) threshold \( I_p \geq \theta_p \)
12. for blob in \( I_b \)
13. \( \quad \) if area of blob \( \geq \theta_a \) then
14. \( \quad \quad \) Add blob to \( L \)
15. end
16. end
17. end

We search the parameter space by an increment of 5 for the number of feature maps (from 5 to 50), and by an increment of 10 for the number of hidden neurons (from 10 to 140). Because of time constraints, we performed only 10 runs per each combination of parameters. This is not enough to pinpoint a definite winner and report statistically significant comparisons for the different possible architectures. Moreover the use of early-stopping prevents overfitting. We can only conclude that larger layers than those listed above do not improve generalization performance on our dataset.

A. Influence of the negative training windows

To study the influence of the negative training examples, we compared the performance of CNNs trained with randomly selected background windows and “hard to classify” windows. These negative windows are hard in the sense that they were misclassified previously by another CNN.

During a run, the set \( L \) of large images is partitioned randomly into a training set \( L_{train} \) and a test set \( L_{test} \). From \( L_{train} \), we build a set \( W_{train} \) of training (and validation) windows. From \( L_{test} \), we build a set \( W_{test} \) of test windows.

For each partition of \( L \), we compare two CNNs trained on the same positive windows from \( W_{train} \), and tested on the windows of \( W_{test} \). The first CNN \( \mathbb{N}_r \) is trained with negative windows drawn randomly from the background, whereas the second CNN \( \mathbb{N}_b \) is trained with some hard negative windows.

Both CNNs are trained from scratch for the same number of epochs (1000). The only difference is the training set of negative windows. Each circle in Figure 8 corresponds to a paired CNN \((\mathbb{N}_r, \mathbb{N}_b)\). The horizontal coordinate of the circle is the generalization error of \( \mathbb{N}_r \). The vertical coordinate of the circle is the generalization error of \( \mathbb{N}_b \). We observe on Figure 8 that \( \mathbb{N}_b \) performs generally better than \( \mathbb{N}_r \).
Fig. 6. A sample of challenging windows (28 × 28 pixels).

Fig. 7. Algorithm 2 computes for each pixel the confidence that this pixel is at the center of a window containing a dugong (center image). The blobs that are too small or with low average confidence are filtered out (right image).
To evaluate the statistical significance of this observation, we applied the Wilcoxon Signed Rank Test [25]. This is a non-parametric statistical test for hypothesis testing on medians. The null hypothesis is that there is no significant difference in the generalization error between the two types of training. The average test error is 3.84 for $N_r$ and 2.59 for $N_A$. The p-value of the test on our 17 runs is $1.14 \times 10^{-4}$. That is the probability of observing the given result, or one more extreme, by chance if the null hypothesis is true is less than $1.14 \times 10^{-4}$. The null hypothesis is rejected. Therefore we can conclude that the difference is statistically significant, and that retraining on hard to classify negative examples is beneficial. The algorithmic template outlined in Algorithm 1 is applicable to any parameterized classifier (not restricted to CNNs).

The authors would like to thank the staff of QUT High Performance Computing and Research Support Center for their support.

**ACKNOWLEDGMENT**

Our study demonstrates that CNNs are a suitable tool for the task of detecting marine mammals in aerial images. In particular, we showed that CNNs outperform classification methods based on hand-crafted features on low resolution images ($28 \times 28$ pixels).

The second contribution of this paper is a machine learning meta-algorithm (Algorithm 1) that automatically creates useful negative examples from a collection of marked images, where the marks consist of bounding boxes around objects of interest (in our case, the objects of interest are dugongs). Like AdaBoost [26], our meta-algorithm modifies its training set, but it is cruder than boosting methods as it does not take a weighted average of the classifiers it trained. It simply returns the last trained classifier as its final output. In practice, one generation is enough when using classifiers with a large number of parameters like CNNs.

In future work, we plan to obtain more labeled data in order to train CNN on larger inputs ($100 \times 100$ windows). We plan to obtain labeled images that have been manually reviewed by experts. The new images are larger than the original dataset (twice the size) but the dugongs should conform to a similar pixel size. By extending our dataset we will be able to train our CNN with a wider range of environmental conditions, which vary the background and appearance of dugongs in the images.

We also plan to replace the sliding window approach sketched in Algorithm 2 with a CNN that takes the whole large image as an input and efficiently computes the bounding box locations. Recent work [27] on CNNs shows that large computational savings can be achieved this way.

**REFERENCES**


