Deep Learning for Animal Recognition

Emmanuel Okafor

ISBN printed version: 978-94-034-1460-7
ISBN electronic version: 978-94-034-1459-1
Cover Design: Emmanuel Okafor
Printed by: HAVEKA

This research was supported by University of Groningen, the Netherlands and Ahmadu Bello University, Nigeria.

Deep Learning for Animal Recognition

PhD thesis

to obtain the degree of PhD at the
University of Groningen
on the authority of the
Rector Magnificus, Prof. E. Sterken,
and in accordance with
the decision by the College of Deans.

This thesis will be defended in public on

Friday 8 March 2019 at 14:30 hours

by

Emmanuel Okafor

born on 25 May 1986
in Zaria, Nigeria

Supervisor
Prof. L.R.B. Schomaker

Co-supervisor
Dr. M.A. Wiering

Assessment committee
Prof. R.C. Veltkamp
Prof. A. Sperduti
Prof. L.V.E. Koopmans

CONTENTS

ACRONYMS

BOW	Bag of Visual Words
CNN	Convolutional Neural Network
DA	Data Augmentation
Faster-RCNN	Faster Region-based Convolutional Neural Networks
HOG	Histogram of Oriented Gradients
***k*-NN**	*k*-Nearest Neighbor
MSR	Multi Scale Retinex
MSRCR	Multi Scale Retinex with Color Restoration
NAGD	Nesterov Accelerated Gradient Descent
RBF	Radial Basis Function
ROT-DA	Rotation-matrix Data Augmentation
SGD	Stochastic Gradient Descent
SSD	Single Shot Multi-Box-Detector
SVM	Support Vector Machine
UAV	Unmanned Aerial Vehicle

1 INTRODUCTION

Wildlife and domestic monitoring of animals is an interesting area of research. This interest arises due to the increasing threat of animal rustling in some African countries and endangered wildlife animals in some European countries and other parts of the world. Hence to best protect and monitor the livestock or the conservation of wild animals, there is a need to deploy technological systems with the prowess to combat the above stated problems. One such technological system is the use of neural network systems, or computer vision techniques combined with machine learning algorithms to deal with these problems. This thesis concentrates on the use of computer vision techniques, machine learning and deep learning techniques for performing recognition, detection, or a combination of both tasks.

The main problem is to determine how the two broad techniques can be used to extract features from images and then predict the corresponding image labels. This problem even becomes more pronounced when objects or animals exhibit similarities in appearance or background information. The use of classical computer vision methods to approach these problems could involve tedious feature engineering, and cannot easily be adapted or transferred to new application domains because they are domain specific. To address these challenges, the emergence of deep learning (Krizhevsky et al., 2012) provides several learning possibilities for instance: use of transfer learning through which pretrained weights from one domain can be transferred or adapted to another application domain. The use of deep learning has recorded a lot of success in several tasks such as object classification (Szegedy et al., 2015; He et al., 2016b), detection (Liu et al., 2016b; Ren et al., 2015), and segmentation (Chen et al., 2018). The success of most of the deep learning methods relies on training deep neural networks on large image datasets.

This thesis aims to achieve the following objectives: we extended the research of deep learning on small datasets with a limited amount of images. Additionally, we explore the concept of reduced deep neural network architectures compared to standard architectures, and classical computer vision methods. To further enhance recognition system accuracies on either aerial or still views, we propose a rotation-matrix data-augmentation (DA) method and a hybrid variant that combines rotation-matrix and color-constancy as another approach to data-augmentation. The latter aids the recognition system to be robust to illumination variance. Furthermore, the study also attempts to explore the benefits of different color spaces for deep learning. Finally, we want to investigate neural network based detection techniques for recognizing and detecting instances of a specific animal.

The earlier mentioned broad recognition systems are examined on images from several datasets: still images (Wild-Anim dataset, Bird-600 dataset, Croatia-fish dataset), aerial images (UAV dataset containing cow and non-cow images), segmented images (Animal-shape and MPEG-7 datasets), and images from a rescue center (Badger dataset). For this aim, the use of classical methods and customized neural network architectures are used for feature extraction. Consequently, the supervised learning algorithm is used for detecting or classifying an image depending on the dataset under study. Additionally the study attempts to propose novel approaches to data augmentation (rotation matrix algorithm alone, or a hybrid variant that factors color constancy) for either enhancing an image or increasing the number of images during training of a given network. Overall the comparison of classical approaches to deep learning methods on the Wild-Anim or Aerial datasets show that the latter method always yields surpassing performance when compared to the former. Moreover, it is important to use the proposed data-augmentation algorithms to obtain improved recognition performances.

1.1 Animal Recognition

Animal Recognition

Animal recognition is an area of research that involves the use of a computer vision algorithm for extracting features from an image or video, and then uses machine learning algorithms for predicting the labels of a given image. The study of animal recognition presents several societal benefits: 1) It allows the monitoring and conservation of wildlife animals especially in an environment where some animals are on the verge of extinction. 2) It also provides the public an important tool to inspect and monitor animal population changes over a period of time. 3) It allows biologists and ecologists to better understand the impact of the animal population to their environment (Wilber et al., 2013).

A lot of previous research on the animal recognition task has employed a classical approach to deal with the classification of different instances of images within a given dataset. The research by Guilford et al. (2009) explores the use of supervised and unsupervised learning algorithms for classifying bird activities based on simple properties obtained from immersion data. An extension of this research was investigated by Dickinson et al. (2010); they developed an automatic visual system for monitoring nesting seabirds. Another improvement in the recognition of seabirds research (Qing et al., 2011) is the use of a boosted combination of the histogram of orientated gradients (HOG) and local binary patterns (LBP) (Pietikäinen, 2010) for extracting features before classification. The research by Wilber et al. (2013) designed a classical approach to recognizing animals in the desert, by using LBP and SIFT (Lowe, 2004) for feature extraction and used a one-vs-all support vector machine (SVM) for classification. The research by Lazebnik et al. (2005) examines a probabilistic part-based method for texture and object recognition of birds. One trending approach that often surpasses most classical methods is the convolutional neural network (CNN) (Schmidhuber, 2015). The research by Jaeger et al. (2015) combined a CNN with a linear SVM classifier for recognizing fishes.

CNNs are part of deep learning methods and will play a central role in this dissertation.

Animal Detection and Localization

The main difference between the previously explained animal recognition and detection is that the latter involves accurately classifying and finding the location of the animal in an image. The use of detection algorithms is important in computer vision systems as these aid segmenting or localizing the region of interest from an image. Shallow approaches to detection adopted the background subtraction technique (Elgammal et al., 2000; Chen, 2009), other background differencing variants (Liu and Hou, 2012; Liu et al., 2016a; Sengar and Mukhopadhyay, 2017), and optimal flow (Zhou and Zhang, 2005) are algorithms for detecting objects of interest in motion. The authors in (Porto et al., 2013) developed a system that comprises of a multi-camera video-recording system, the software component uses the Viola-Jones (Viola and Jones, 2004) algorithm for detecting behaviors of lying cows. In this dissertation to determine where an animal is in an image, it is crucial to employ the neural network based detection algorithms to identify the animal location within an image.

Image Enhancement

Most computer vision or deep learning methods rely on enhanced images to obtain improved detection or recognition performances. Image enhancement algorithms aim at modifying content information or attributes present in an image to make it suitable for specific application purposes. Image enhancement techniques (Maini and Aggarwal, 2010) can be broadly grouped into two domains: spatial domain methods manipulate pixels in an image and frequency domain methods transform an image into the Fourier transform domain. Most data-augmentation methods such as: color-casting, rotation, flipping, cropping, and scaling use the spatial domain for modifying original image contents. Proper image enhancement can help recognition systems to obtain better results.

The three broad topics of this dissertation as discussed above can be grouped into five objectives. The next sections briefly discuss the objectives, the contributions and their respective research questions for this thesis.

1.2 Objectives of the Thesis

This dissertation examines animal recognition and detection systems. The objectives of the research can be grouped into three broad categories: Use of compact neural networks, image enhancement, and adequate localization of objects of interest. The five detailed objectives of this thesis are described below:

Firstly, to analyze the best image recognition method when there are not many images. Most of the datasets in this application domain contain a relatively small amount of images. However, the use of existing neural network architectures requires a considerable amount of neurons, network parameters, and needs massive training data and long training times. Therefore we propose compact neural network architectures with fewer network parameters used during training of the network which lowers computational cost, while retaining a suitable classification performance.

Secondly, to handle rotation invariance in unmanned aerial vehicle (UAV) images without creating too many images. The orientation of the path of flight and the orientations of the target objects that is, the animals will be random (haphazard). Therefore, we propose a rotation matrix algorithm as a novel method of data augmentation (DA). Conventional DA techniques transform input data and increase the amount of training data when there exist insufficient data, with an aim to obtain better classification results. The new DA method is useful for enhancing the pixel information in an image. Additionally, the new DA method does not require an increase in the amount of images during training of the network compared to the conventional DA approaches. The DA methods combined with pretrained instances of the used reduced neural network obtain high classification scores.

Thirdly, to develop a recognition system that is robust to illumination variances due to varying daytime light conditions (day or night) and weather direction of sunlight. For this purpose, we developed a hybrid variant of the rotation-matrix data augmentation that combines rotation-matrix and color constancy as another method for DA. The proposed technique can be used to increase the number of training images especially when there exists an insufficient amount of images within a dataset. Another merit of the proposed method is that it can enhance the illumination quality of a blurred image. Additionally, an appropriate selection of grid resolution and angular bounds can aid the pretrained instance of the used reduced neural network to obtain high classification scores.

Furthermore, we analyze how important the use of color spaces is in deep learning. For this, we construct a color conversion algorithm that has the potential to transform a natural (RGB) or black and white (BW) images to four other color spaces. Then we employed our custom network to access the classification performance on several variants of the used animal datasets.

Lastly, we want to analyze detection algorithms for detecting and recognizing individual instances of badgers. One primary goal is to help biologist (zoologist) who does not have time in developing detection systems, to create a system that can detect and classify instances of the mentioned animal. However, the main problem is that localizing and finding an object of interest in an image is a difficult task within the computer vision domain, especially when there exist high similarities in object appearances. We investigated the use of neural network based detection systems to adequately determine where an instance of an animal is within a given image. The resulting model could be deployed into real-time systems such as a drone or other data-acquisition systems.

1.3 Dissertation Overview

Comparison of classical methods to customized deep learning methods

In Chapter 2 of this thesis, we compare several classical computer vision methods combined with a supervised learning algorithm to customized and existing deep learning techniques for recognizing still images of wild animals. We attempt to answer the following research questions: *Is there any benefit of reducing network neurons from an existing deep learning architecture? How well do reduced neural network architectures perform relative to classical computer vision techniques for classifying wild animals?* To provide a solution to this challenge, we modified existing deep convolutional neural network (CNN) architectures (AlexNet and GoogleNet) by reducing the number of neurons in each layer of the fully-connected layer (AlexNet) and each layer in the last inception module of the GoogleNet architecture (with an exemption of the first layer). The new architectures use fewer neurons and reduce computational costs during training of the network models. Additionally, the proposed network architectures present almost similar performance levels when compared to existing networks. Moreover, we compared these deep learning architectures to classical techniques: variants of the bag of visual words (BOW) alone or BOW with the histogram of orientation gradients (HOG-BOW) with a regularized support vector machine (SVM). The results show that most of the deep neural network methods either in their existing or reduced forms yield performances that surpass the classical approaches when examined on our relatively small dataset.

Rotation-matrix data augmentation on UAV images

We enhanced aerial images of cows and non-cow backgrounds before applying recognition systems. In Chapter 3, we examine the following research questions: *Can the transformation of aerial images to rotation-matrix images enhance recognition systems to obtain high predictive scores? How well do a more shallow depth neural network architecture or classical methods classify rotation-matrix data-augmentation compared to non-rotation-matrix (original) images?* To provide a possible solution to the stated research questions, we propose a novel rotation matrix data-augmentation technique that transforms a train or test image into a novel single image with multiple randomly rotated copies of the input image. To combine the different rotated images, the proposed method puts them in a grid and adds realistic background pixels to glue them together. This approach presents some advantages: 1) It provides more informative images which may aid to yield higher accuracies, 2) It does not require an increase in the number of training images compared to other conventional data-augmentation methods. The use of fine-tuned CNN models with the proposed data-augmentation technique leads to significantly better results than the classical approaches. The study again shows the relevance of reducing the depth of neural network architectures.

Unification of rotation matrix and color constancy

Previous approaches to data augmentation use cropping, rotation, illumination, scaling, and color casting for creating more training images. Chapter 4 of this thesis attempts to answer the following research questions. *Can unifying the rotation matrix and color constancy algorithms operated on different animal images be considered as a promising method of data-augmentation? What role do an appropriate choice of selection of grid resolution and angular bounds play for the proposed data augmentation (DA) technique?* We propose the combination of both color-constancy and rotation matrix algorithm for transforming an input image. Since the recommended approach results in an increase of the number of training images, it can be considered as a method of data-augmentation similar to the conventional approach. A merit of the proposed DA method is that it enhances the color information in an image which could be useful for obtaining higher recognition accuracies. The study further shows that, using finetuned CNNs with an appropriate selection of the grid resolution and angular bounds for the rotation algorithm combined with color-constancy methods yields the highest classification accuracies on most of the used datasets.

Analysis of color spaces for image recognition in deep learning

Several research works have focused on employing machine learning algorithms for classifying natural or Black/White (BW) binary images. Chapter 5 of this thesis attempts to examine the conversion of the two broad kinds of images (natural or BW) into other color spaces before applying a recognition system. *Does the conversion of datasets containing either of the mentioned images or new variants of the images affect the performance of neural networks?* To provide a possible solution to the above research question, we describe the use of different versions of the GoogleNet architecture (finetuned and scratch instances) for investigating the classification performances on different color versions of image datasets. We propose a color conversion algorithm, which presents the following merits: It can transform binary masked (BW) images to images represented in different color spaces (RGB, YCbCr, HSV, Lab), which show marginal CNN classification performance improvements for some of the methods. Additionally, it is an efficient algorithm and easy to implement or use.

Detection and recognition of badgers using deep learning

Chapter 6 deals with the detection and recognition of badgers under varying illumination backgrounds. *Which of the detection neural network algorithms is most suitable for application purposes especially at the deployment phase?* To provide an answer to the research question, we propose the use of several object detection algorithms based on deep neural networks for detecting and recognizing badgers from video data. For this, a comparison is made between two neural network based detectors: SSD (Liu et al., 2016b) and Faster R-CNN (Ren et al., 2015). SSD is combined with the Inception-V2 (Ioffe and Szegedy, 2015) or MobileNet (Howard et al., 2017) as a backbone and the Faster R-CNN detector is combined with either Inception-V2 or Residual networks (He et al., 2016a) with 50 layers (ResNet-50) as feature extractors. Furthermore, we compare the use of two output activation functions: the softmax and sigmoid function. For the experiments, we use several videos recorded with a low-resolution camera. The results show that most of the trained SSD detectors significantly outperform the different variants of the Faster R-CNN detector. All the Faster R-CNN methods are computationally much faster than the SSD techniques for training the system, although for testing SSD is a bit faster. Hence, we suggest that the best found model, SSD-Inception-V2-Softmax, could be improved and deployed into UAVs or thermal acquisition cameras, as this can help to detect badgers in environments where they are endangered.

Finally, Chapter 7 concludes the dissertation and briefly discuss the achieved objectives of this thesis. The chapter also provides areas for future research.

2 CLASSICAL AND DEEP LEARNING METHODS

This chapter addresses the problem of animal recognition and examines the benefit of modifying convolutional neural network (CNN) architectures for this application. To achieve this aim, two broad classical feature extraction methods are compared to deep learning techniques with an overall objective of recognizing animals. For the classical approaches, variants of the bag of visual words (BOW) alone and BOW with the histogram of oriented gradients (HOG-BOW), each using two forms of spatial pooling approaches are applied on two kinds of feature extraction method either using color or gray level intensities. The final feature vectors extracted from these BOW variants combined with an L2 regularized support vector machine (L2-SVM) is used to distinguish between classes of the used dataset. Moreover, we modified existing deep CNN architectures (AlexNet and GoogleNet) by reducing the number of neurons in each layer of the fully-connected layer (AlexNet) and each layer in the last inception module of the GoogleNet architecture with an exemption of the first layer. The CNN was trained using random weights (scratch) and pretrained weights (finetuned). The existing CNN and the modified CNN architectures are compared to the proposed BOW variants on a novel wild-animal dataset (Wild-Anim). The experimental results show that the deep CNN methods significantly outperform the traditional BOW techniques.

This chapter is based on the paper:

Okafor, E., Pawara, P., Karaaba, F., Surinta. O., Codreanu. V., Schomaker, L.R.B., and Wiering, M.A. (2016). Comparative Study Between Deep Learning and Bag of Visual Words for Wild-Animal Recognition. *IEEE Symposium Series on Computational Intelligence (IEEE SSCI), pp. 1-8*

The field of computer vision has the aim to construct intelligent systems that can recognize the semantic content displayed on images. Most research in this field has focused on recognizing faces, objects, scenes, and characters. In this chapter, we describe several techniques that use machine learning and pattern recognition methods to recognize wild animal images, which has gained less attention from the community. The concept of recognition of objects based on variations in image content has gained attention over several decades now, and has lately received an increased interest due to the advance of deep learning techniques (Schmidhuber, 2015). This chapter focuses on different methods from the computer vision community in which deep CNNs, feature descriptors and machine learning algorithms are used to predict labels of animal images.

Some approaches to animal, object and scene recognition have concentrated on the use of color descriptors (Van De Sande et al., 2010; Khan et al., 2013; Sergyán, 2008). Also, the authors in (Khosla et al., 2012) investigated the combination of local and global features for modelling a framework for memorability prediction. In a quest to improve recognition performance, the use of classical image descriptors such as the Bag-of-Visual-Words (BOVW)[1] has been applied to different fields. BOW comes from traditional information retrieval (text) (Salton et al., 1971). The concept of BOW involves the extraction of features (Csurka et al., 2004; Wang and Huang, 2015) and construction of a codebook using an unsupervised learning algorithm such as K-means clustering (Ye et al., 2012), spectral clustering (Passalis and Tefas, 2016), local constrained linear coding for pooling clusters (Wang et al., 2010), and the use of the fast minimum spanning tree (Jothi et al., 2015). Finally, the extraction of feature vectors by the BOW approach can be achieved using a soft assignment scheme (Abdullah et al., 2010) or sparse ensemble learning methods (Tang et al., 2015). Some recent works have used BOW as an input to some hierarchical structures such as weakly supervised deep metric learning (Li and Tang, 2015) and robust structured subspace learning (Li et al., 2015). Moreover, the combination of BOW with the histogram of oriented gradients on grayscale datasets has obtained a very good performance on both handwritten character recognition (Surinta et al., 2015) and face

[1] For convenience, we refer to BOVW as BOW

recognition (Karaaba et al., 2016). In (Coates et al., 2011a), the authors applied BOW on text detection and character recognition on scene images.

However, the concept of BOW has become old fashioned by the recently emerging and successful area of deep learning with neural networks. These learning techniques have been successfully applied to many applications such as human face recognition (Pinto et al., 2011; Parkhi et al., 2015), object recognition (Krizhevsky et al., 2012), handwritten character recognition (LeCun et al., 1989, 1998; Ciresan et al., 2011) and medical image recognition (Shin et al., 2016). The use of deep learning to learn from large datasets has led to the evolution of deep architectures like AlexNet (Krizhevsky et al., 2012), GoogleNet (Szegedy et al., 2015) and Residual Networks (ResNets) (He et al., 2016b).

The BOW method (Csurka et al., 2004) has been a popular and widely used method in the computer vision community. According to (Coates et al., 2011b), the BOW technique outperforms other feature learning algorithms like autoencoders and restricted Boltzmann machines. In addition to the survey on the use of convolutional neural networks, the authors in (Girshick et al., 2014) showed that regions with CNN (R-CNN) features outperform HOG-based deformable part models and feature learning based methods on PASCAL VOC datasets. Also, the authors in (Razavian et al., 2014) demonstrated that CNN augmentation with a support vector machine (SVM) outperforms BOW and other local feature descriptors.

Contributions

In this chapter, an investigation of the performance of 16 different techniques on a novel wild-animal dataset, is proposed. To actualize this aim, the use of existing deep CNN architectures (GoogleNet and AlexNet), the modified versions of the deep CNN (Reduced Fine-tuned and Scratch versions of GoogleNet and AlexNet) and variants of BOW techniques are applied to a novel Wild-Anim dataset. The results show that the modified CNN architectures are competitive when compared to the original deep CNN architectures but require less computing time. This is evident based on the significant decrease of the computational time by 27% and 26% for both the fine-tuned and scratch versions of the AlexNet architectures respectively. Also, we compared the deep CNN architectures to different variants of the BOW approach combined with an SVM with major emphasis on two spatial pooling strategies as well as the use of color information on a Wild-Animal dataset. The results show that the GoogleNet CNN architectures perform best. Furthermore, almost all CNN architectures significantly outperform all BOW variants. The results also show that the BOW method using color information with the max-pooling strategy outperforms the HOG-BOW methods for both gray and color image information on the used dataset for both spatial pooling strategies. This is contrary to the view that HOG-BOW techniques outperform BOW methods, which was shown before in character recognition (Surinta et al., 2015) and facial recognition (Karaaba et al., 2016).

Outline. This chapter is organized in the following way. Section 2.1 briefly explains the basic deep learning processes. Section 2.2 describes the different learning techniques used in the wild animal recognition system. Section 2.4 describes the Wild-Anim dataset that is used in the experiments. The experimental results of the deep learning methods and bag of visual words are presented in Section 2.5. The conclusion and future work are reported in Section 2.6.

2.1 Basic Deep Learning Processes

In order to understand the activities going on in each stage of a deep neural network, below we briefly explain the processes based on some mathematical principles.

Convolution Process: Convolutional layers employ learnable filters which are each convolved with the layer's input to produce feature maps. The feature map $Z^l(x, y, i)$ for neuron i from each convolutional layer l can be computed as:

$$Z^l(x, y, i) = B_i^l + X^{l-1}(x, y, c) * K_i^l(x, y, c) \tag{1}$$

The input to the convolutional neural network can be represented as a tensor X^{l-1} from the previous layer with elements $X(x, y, c)$, denoting the value of the input unit within channel c at row x and column y. The input to the convolution is convolved with the tensor kernel using a bank of filters K_i^l for the current layer with the same number of channels present in

X^{l-1}. Each convolved feature map in a given layer gets its corresponding bias B_i^l added.

Detector Process: This process involves the use of a non-linear activation function such as the Rectified Linear Unit (ReLU) (Krizhevsky et al., 2012) to compute activations of all convolved extracted features. The ReLU is often assigned to the output of each hidden unit in a convolutional layer and the fully connected layers. The output of the ReLU $P^l(x, y, i)$ is calculated using the expression:

$$P^l(x, y, i) = max(0, Z^l(x, y, i)) \tag{2}$$

Normalization Process: In this process, local response normalization is used for normalizing the output of the ReLU (Krizhevsky et al., 2012; Vedaldi and Lenc, 2015). The role of the local response normalization is assumed to yield better generalization and introduces non-linearity that is absent in the right hand side of the ReLU responses. The local response normalization (Stutz, 2014) can be computed as:

$$Q^l(x, y, i) = P^l(x, y, i)\left(\gamma + \alpha \sum_{j \in M^l} (P^l(x, y, j))^2\right)^{-\beta} \tag{3}$$

where $Q^l(x, y, i)$ computes the response of the normalized activity from the ReLU output $P^l(x, y, i)$. This is done by multiplying the output with an inverse sum of squares plus an offset γ for all ReLU outputs within a layer l. The normalization is local over the feature map M^l. We employed the same hyper-parameter setting as in (Krizhevsky et al., 2012) with the following constant variables: $\gamma = 2$, $\alpha = 10^{-4}$ and $\beta = 0.75$.

Spatial Pooling Process: In this process, two spatial pooling approaches are employed in the two CNN architectures used in the experiments.

1. Max-Pooling: The max-pooling operator computes the maximum response of each feature channel obtained from the normalized output. A max-pooling operator can be expressed as:

$$R^l(\bar{x}, \bar{y}, i) = max_{x, y \in M(\bar{x}, \bar{y}, l)} Q^l(x, y, i) \tag{4}$$

Where $(\bar{x}, \bar{y})$ is the mean image position of the positions (x, y) inside $M(\bar{x}, \bar{y}, l)$ that denotes the shape of the pooling layer, and $R^l(x, y, i)$ is the result of the spatial pooling of the convolutional layers.

2. Average-Pooling: The average-pooling operator computes the mean response of each feature channel obtained from the normalized output. An average-pooling operator can be expressed as:

$$R^l(\bar{x}, \bar{y}, i) = \frac{\sum_{x,y \in M(\bar{x},\bar{y},l)} Q^l(x, y, i)}{|M(\bar{x}, \bar{y}, l)|} \tag{5}$$

Regularization Process: In order to reduce over-fitting in the network, the use of the dropout (Krizhevsky et al., 2012) regularization scheme is applied to the output of the spatial pooling layer or mid-level fully-connected layers. Dropout as a regularization technique can aid to prevent complex co-adaptations on a training data. The dropout phenomenon refers to an act of dropping hidden nodes in a neural network based on a defined probability.

Classification Process: In this process, the probability of the class labels from the output of the fully connected layer is computed using the softmax activation function. The softmax activation function (Goodfellow et al., 2016) computes the probabilities of the multi-class labels using the sum of weighted inputs from the previous layer and is used in the learning process:

$$y_d = \frac{\exp(x_d)}{\sum_{d=1}^{D} exp(x_d)} \tag{6}$$

where y_d is the output of the softmax activation function for class d, x_d is the summed input of output unit d in the final output layer of the fully connected network and D is the total number of classes.

Often, the classification process employs the use of the top-K classification error for computing the errors on the testset. The top-K loss is zero if target class d is within the top K ranked scores (Vedaldi and Lenc, 2015):

$$L(y, d) = 0[|\{k : y_k \geq y_d\}| < K] \tag{7}$$

The top-K loss is one for an example, if:

$$L(y, d) = 1[|\{k : y_k \geq y_d\}| \geq K] \tag{8}$$

Where y_d are the final outputs of the CNN. We report results of the top-1 error accuracy in all the experiments.

2.2 Learning Methods

This section discusses both deep learning using convolutional neural networks (CNNs) and variants of bag of visual words combined with a Support Vector Machine (SVM) to deal with the wild animal dataset. We will make use of two deep CNN architectures, AlexNet and GoogleNet, and modify them. We will now explain these architectures and the modifications, that results in 8 different deep learning architectures.

2.2.1 AlexNet Architecture

The AlexNet model, initially proposed in (Krizhevsky et al., 2012), outperformed the non-deep learning methods for the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012. AlexNet consists of five convolution layers, three pooling layers, and three fully connected layers with approximately 60 million trainable parameters. This chapter explores the use of both original and reduced versions of both scratch and fine-tuned AlexNet models on the Wild-Anim dataset. Our experimental procedure is similar to that of (Krizhevsky et al., 2012), that applied the stochastic gradient descent update rule with momentum, which is expressed as:

$$u_{i+1} = \mu u_i - \alpha_L \left(\delta W_i + \left(\frac{\partial L}{\partial W_i} \right)_{D_i} \right) \tag{9}$$

$$W_{i+1} = W_i + u_{i+1} \tag{10}$$

where W_i are the weights of the CNN, u_i is the weight change, L is the cross-entropy loss function uses the softmax activation for a given class, μ

is the momentum term, α_L is the learning rate, δ is the value for weight decay, i is the iteration number, D_i is the batch over index iteration i and $\left(\frac{\partial L}{\partial W_i}\right)$ computes the mean over the i^{th} batch D_i of the derivative in the objective function with respect to W_i. We will now briefly explain the AlexNet architecture models.

Scratch AlexNet: We will first train the AlexNet architecture from scratch on the train-validation sets based on 5 different random shuffles of the used dataset in order to obtain models that can be used to evaluate on the test sets. The experimental settings are as follows; crop size 227 × 227, momentum 0.9, weight decay 5×10^{-4}, test iteration of the solver is 10, batch size of training 10, test interval 100, base learning rate 1×10^{-3}, learning policy is step with a step-size of 3×10^{4}, a dropout of 0.5, gamma 0.1, with a maximum number of iterations of 30000, which generates a snapshot model after every 1000 iterations. In this architecture only the max-pooling strategy is used in the spatial pooling layers. This setting is for the Original Scratch AlexNet (OS-AlexNet) model which has 4096 neurons in each of the fully-connected layers (FC6 and FC7) except in the last layer FC8 in which the number of output neurons is equal to the number of classes within the dataset. We modified the OS-AlexNet model by reducing the number of neurons per fully-connected layer (FC6 and FC7) to 512, since this modification results in less demand on the computer memory usage and speeds up the use of this architecture. The block diagram in Figure. 1 illustrates the modified version of the AlexNet architecture. The choice of 512 neurons in each of the fully connected layers is because it gives the best results after several experiments with different numbers of neurons on the used dataset.

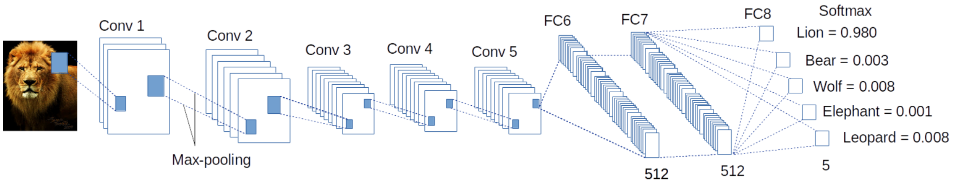


Figure 1: Block diagram of modified AlexNet architecture with reduction in neurons in the fully connected layers.

Fine-tuned AlexNet: This version of the architecture relies on the weights that are initialized by a pre-trained network. The pre-trained network is trained on a subset of ImageNet (ImageNet Large-Scale Visual Recognition Challenge (ILSVRC)) (Krizhevsky et al., 2012). This version of the dataset consists of a minimum of 1000 images for each of the 1000 classes. The dataset is roughly divided into 1.2 million training images, 50,000 validation images, and 150,000 testing images. Although the ILSVRC ImageNet dataset has some categories of images which also occur in the used dataset, the datasets contain different images.

The Original Fined-tuned AlexNet (OFT-AlexNet) and Reduced Fine-tuned AlexNet (RFT-AlexNet) require a pre-trained CNN architecture model. The pre-trained network of the AlexNet architecture was constructed by training on the ILSVRC ImageNet dataset. We maintain the same experimental settings as discussed earlier. One exception is that the maximum number of iterations is reduced to 10000 (10 snapshots) with a step size of 10000 using a fixed learning rate of 0.001. We note that all the experiments were carried out using the Caffe platform on a Ge-Force GTX 960 GPU model.

2.2.2 GoogleNet Architecture

GoogleNet (Szegedy et al., 2015) is a famous deep learning architecture that won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2014. This architecture is inspired by incorporating several inception modules (Arora et al., 2014) which allows the stacking or concatenation of filters of different dimensions and sizes into a single new filter (Shin et al., 2016). This architecture consists of some outer convolutional and pooling layers, three classifiers (two intermediate and one main) with a regularization dropout of 0.7, 0.7 and 0.4 placed after the intermediate fully connected-layer or main average pooling layer (at the top of the ConvNet) respectively. This dropout helps to avoid overfitting during training. Furthermore, this architecture has nine inception layers. In each inception layer, there exist six convolution layers with different dimensions of filters and one pooling layer. The architecture uses both average and max-pooling strategies for creating smaller feature maps. We describe briefly in the following subsections the various instances of GoogleNet used in our study.

Scratch GoogleNet: The Scratch GoogleNet architecture does not rely on any pre-trained CNN model. The experimental settings are as follows; crop size 224 × 224, momentum 0.9, weight decay 2×10^{-4}, test iteration 10, batch size 10, test interval 100, base learning rate 1×10^{-3}, step-size of 3×10^{4}, interval display 40, average loss 40, power 0.5, gamma 0.1 and a maximum number of iterations of 30000 (30 snapshots). The number of output neurons fed to the three classifiers of this architecture is equal to the number of classes present in our dataset. The GoogleNet architecture uses both the max-pooling and average pooling strategies in different spatial pooling layers.

This setting is for Original Scratch GoogleNet (OS-GoogleNet) with the last inception layer which contains a max-pooling layer and six convolutional layers. The number of filters (neurons) in each layer within the last inception layer is as follows: 384, 192, 384, 48, 128 and 128 respectively. In the Reduced Scratch GoogleNet (RS-GoogleNet) the last inception layer of the OS-GoogleNet is modified to contain the following numbers of filters in each of the convolutional layers in the last inception layer: 24, 24, 24, 16 and 16 respectively, except for the first layer which has 384 filters. The block diagram in Figure 2 illustrates the modification in the last inception layer of the GoogleNet architecture.

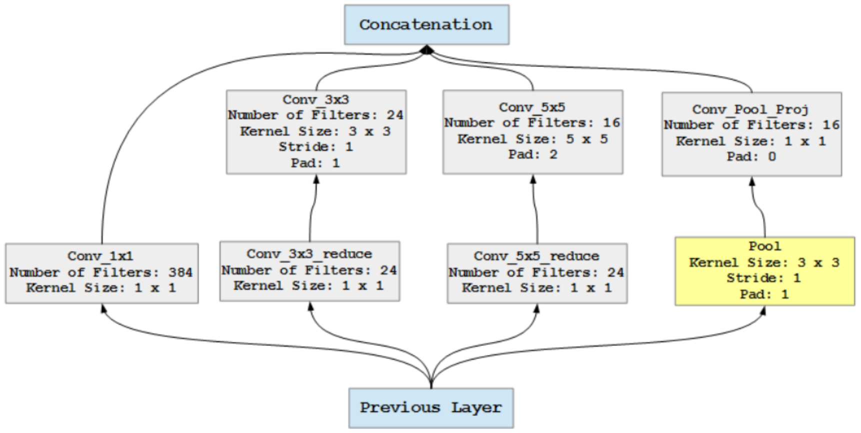


Figure 2: Block diagram showing our modification of the number of output filters for each convolution within the last inception layer of the GoogleNet architecture.

Fine-tuned GoogleNet: The Original Fined-tuned GoogleNet (OFT-GoogleNet) and Reduced Fine-tuned GoogleNet (RFT-GoogleNet) require a pre-trained CNN architecture model. The pre-trained network of the GoogleNet relies on the ILSVRC dataset that was explained in the paragraph about the AlexNet architecture. We maintain the same experimental settings as discussed earlier, except that the maximum number of iterations is reduced to 10000 (10 snapshots) with a step size of 10000. We used a fixed learning rate of 1×10^{-3}.

2.2.3 Variants of Bag of Visual Words (BOW) with SVM

In this subsection, we describe two major kinds of BOW models.

Bag of Visual Words with Image Pixel Intensity: This technique uses the extraction of patches from the training data based on the image pixel intensities to construct a codebook (Ye et al., 2012) using K-means clustering. The diagram in Figure. 3 shows a description of this technique. The steps involved in setting up BOW consist of three processes which we will explain now.

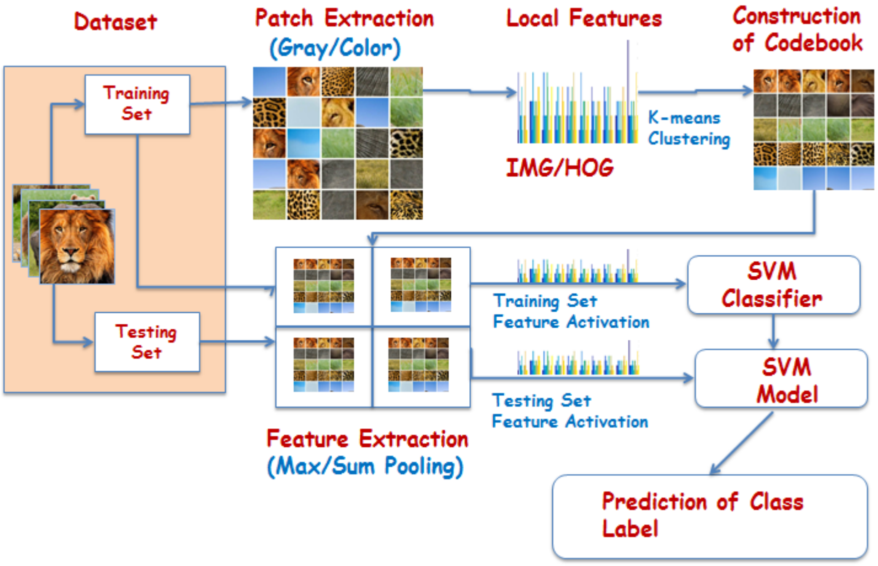

Dataset
Training Set
Testing Set
Patch Extraction (Gray/Color)
Local Features
IMG/HOG
K-means Clustering
Construction of Codebook
Training Set Feature Activation
Testing Set Feature Activation
Feature Extraction (Max/Sum Pooling)
SVM Classifier
SVM Model
Prediction of Class Label

Figure 3: Description of Bag of Visual Words combined with an SVM

Extracting Patches from the Training Data: The images are divided into a set of sub-image patches X that is extracted randomly from unlabelled training images, $X = \{x_1, x_2, x_3, ...x_N\}$ where N is the number of random patches and $x_k \in \Re^t$ is some patch extracted from the training images. The size of the patches is described with $t = p \times p$ pixels. In this experiment, we used $p = 9$, which implies that 81 pixels were used in a patch.

Construction of the codebook: The codebook is constructed by applying K-means clustering on the feature vectors consisting of pixel intensity information which is contained in each patch. This is achieved by clustering the vectors obtained from the random selection of the patches. Let $C = \{c_1, c_2, c_3, ...c_k\}$, with $c_i \in \Re^t$ represent the codebook (Ye et al., 2012), where k is the number of centroids. In the preliminary experiments, we used randomly selected patches to compute the codebook. The final choice was the use of 100,000 patches, because this extracts most information from the animal dataset and has a good trade-off in computational time when compared to a larger number of patches.

Feature Extraction: the soft assignment coding method from (Coates et al., 2011b) is used to create the feature vectors for both training and testing images. The activity of each cluster given all feature vectors x_t from all patches in an image is computed using the equation:

$$i_k(x) = \sum_t \max\{0, \mu(x_t) - q_k(x_t)\} \tag{11}$$

where $q_k(x_t) = ||x_t - c_k||_2$ and $\mu(x_t)$ is the mean of the elements of this distance measure over the centroids c_k (Coates et al., 2011b). We consider two spatial pooling approaches. An image is divided into four quadrants and the activities of each cluster for each patch in a quadrant are summed up. The spatial pooling approach that is described in Equation 11 is referred to as *Sum-Pooling*. While the second pooling approach, *Max-Pooling*, is the computation of the maximum cluster activity given a feature vector x_t from all patches which are in an image and can be described using the expression:

$$i_k(x) = \max_t\{\max\{0, \mu(x_t) - q_k(x_t)\}\} \tag{12}$$

The patches of testing and training images are extracted using a sliding window. Because we use a stride of 1 pixel, the window size of 9 × 9 pixels and the used image resolution is 250 × 250 pixels, the method extracts 58564 patches from each image. These patches, with the initial random patch extraction and the number of clusters are used for computing the cluster activations using Equation 11 or Equation 12. The feature vector size is $K \times 4$ and since we chose K to be 600 clusters, the feature vector of BOW has 2,400 dimensions.

Bag of Visual Words with Histogram of Oriented Gradients (HOG-BOW): The HOG-BOW method is used to compute feature vectors from patches based on the HOG descriptor (Dalal and Triggs, 2005). The patches are given to the Histogram of Oriented Gradients (HOG) descriptor and the extracted feature vectors are used to calculate the codebook as well as the cluster activities. In order to compute the HOG feature vector (Junior et al., 2009; Takahashi et al., 2014), the HOG descriptor divides each patch into smaller regions known as blocks, $\eta \times \eta$. The HOG descriptor computes two gradients (horizontal gradient h_x and vertical gradient h_y) with respect to every coordinate x, y of an image using and a simple edge detector (kernel gradient detector) (Arróspide et al., 2013). The gradients are computed using:

$$h_x = W(x+1,y) - W(x-1,y) \tag{13}$$

$$h_y = W(x,y+1) - W(x,y-1) \tag{14}$$

where $W(x,y)$ is the intensity value of the coordinate x, y. The magnitude $A(x,y)$ and the orientation $\alpha(x,y)$ are computed as:

$$A(x,y) = \sqrt{h_x^2 + h_y^2} \tag{15}$$

$$\alpha(x,y) = tan^{-1}\left(\frac{h_y}{h_x}\right) \tag{16}$$

The image gradient orientations within each block are weighted into a specified number of orientation bins β, making up the histogram. Finally, L2 normalization is applied to the sum of bin values of the HOG feature vectors (Dalal and Triggs, 2005). In the preliminary experiments, we found the best HOG parameters to use are 25 rectangular blocks ($\eta = 5$) and 8 orientation bins to compute the feature vectors from each patch. In the HOG-BOW experiment the best found patch size is 15×15 pixels. We also modified the HOG-BOW algorithm such that it can process both gray and color information from the patches in the used dataset. In both BOW and HOG-BOW the color information from the patches of an image is computed by concatenation of each of the three channels that makes up the RGB color space for each of the extracted patches. In the same vein as in BOW, HOG-BOW employs 600 centroids and both sum-pooling and max-pooling were applied to the four quadrants on the codebook based on either gray or color images in the used dataset. The HOG-BOW method results in 2,400 dimensional feature vectors.

Finally, the final feature vector from both BOW and HOG-BOW are fed into the regularized linear L2-SVM classifier which predicts the class labels of the Wild-Anim images. We adopted the one-vs-all approach. In a linear multi-class SVM, the output $z_k(x)$ of the k-th class is computed as:

$$z_k(x) = w_k^T i(x) + b_k \tag{17}$$

where $i(x) \in \Re^n$ are the input vectors constructed by the BOW variants from an image x. The linear classifier for class k is trained to output a weight vector w_k with a bias value b_k.

The predicted output class label for an image x (Tang, 2013) is computed using:

$$arg \max_k (z_k(x)) \tag{18}$$

We use the regularized L2-SVM (Fan et al., 2008) for which the primal objective function is given by:

$$\min_w \frac{1}{2} w^T w + C \sum_{i=1}^{n} \left(\max\left(0, 1 - y_i z_k(x)\right)\right)^2 \tag{19}$$

where $y_i = \{1, -1\}$, $y_i = 1$ if x_i belongs to the target class of the k-th classifier, and $y_i = -1$ if x_i does not belong to the target class. C is the penalty parameter.

2.3 Find Optimal Hyperparameter Values

The success of deep learning and machine learning, in general, depends on so-called hyperparameters that control the learning algorithms. The exact value of a hyperparameter has large consequences on the performance of the algorithm. The selection of the optimal value is usually done using a multi-dimensional grid search. For each dimension, given a minimum, a maximum and a step size, the performance is evaluated on a validation set. An exhaustive grid search in high-resolution parameter space is very time-consuming. Where possible, we applied such optimization approaches, for tuning the C parameter of the support vector machine algorithm in the range $2 \leq C \leq 512$, for $C = 2^n$, where $n = \{1, 2, ...9\}$. The best-found C value is 16 (this dissertation, pp. 29). In other cases time and computing resources were limited. In such a case we used optimal parameter values found in the literature (Krizhevsky et al., 2012). It is undeniable that the search for optimal hyperparameter values in machine learning is sensitive and plays an important role. In autonomous machine learning, it would be required to have a self-learning system that can optimize hyperparameters to prevent optimizing for different datasets.

2.4 Animal Dataset and Pre-Processing

In this section our novel dataset and preprocessing steps for the experiments will be described.

2.4.1 Wild-Anim Dataset

We collected a novel dataset by downloading images of animals from Flicker. The dataset is called Wild-Anim derived from wild animals. This dataset consists of a total of 5,000 images and consists of 5 classes, namely; Bear, Elephant, Leopard, Lion, and Wolf. The dataset is processed by automatic labelling and then was normalized to 250 × 250 pixels introducing slightly anamorphic distortions. All images in this dataset are in RGB color space. A sample of the images in the used dataset is shown in Figure. 4. After collecting the dataset, we noticed that ImageNet also contains the same classes.

Therefore, before carrying out our experiments we carefully examined that there is no image overlap between our dataset and that of the ILSVRC ImageNet dataset. So, although the ILSVRC ImageNet dataset has some categories of images which are used in our dataset, it contains different images. We initially trained on the entire dataset with the use of a local feature descriptor (HOG-BOW). We recorded a good performance, but the drawback was that it took approximately two days to complete the computation. In order to mitigate this computational time challenge, we used Deep-CNN which turns-out to be very viable, because it requires less computing time to produce an outstanding result since it runs on a

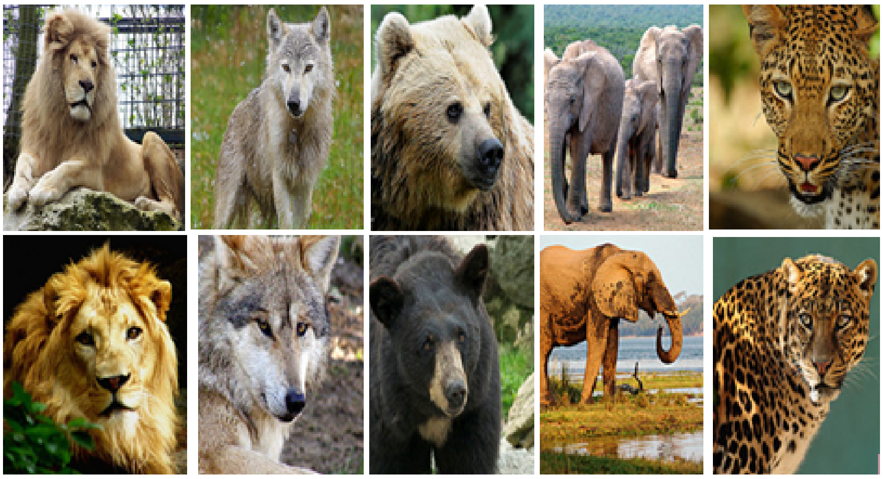


Figure 4: Samples of the images in the Wild-Anim Dataset, from left column to right column: lion, wolf, bear, elephant and leopard.

GPU. This is evident based on the small sample experiment conducted on a 20% subset of our dataset which contains 1000 images. We conducted two kinds of experiments on the 1000 images; 1) On BOW variants, the subsets are randomly partitioned into two basic entities in the ratio 0.9:0.1 for the training set and testing set, respectively. 2) In the CNN approach, we partitioned the dataset into the ratio 0.8:0.1:0.1 for the training set, validation set and testing set respectively. The Deep CNN techniques use an overall computing time for the complete experiment with 5 runs between $0.22 \leq t \leq 2.1$ hours. Of course, this reduction is mainly caused by the used software, where the Caffe framework uses GPU computing, and does not imply that deep CNNs are in general faster than the BOW method. The exact duration depends on the experimental settings of either fine-tuned or scratched versions of the CNN architectures under study. AlexNet is also faster than GoogleNet. For the BOW variants the computing time for an entire experiment is between $0.65 \leq t \leq 26$ hours. In the experiments, five different random shuffles of this subset of 1000 images are used to carry out 5 random-fold cross validation.

2.5 Results

All results in this section are based on 5-fold cross validation. We compute both the mean precision and standard deviation for evaluating the test performance of the Deep CNN architectures and the variants of BOW on our dataset.

2.5.1 Evaluation on the Wild-Anim Dataset

The MATLAB programming platform is used to carry out experiments with the BOW variants. We initially adopt a grid search approach to fine-tune the C parameter in order to determine the best choice of C in the linear L2-SVM algorithm. We finally used $C = 16$ for both kinds of local feature descriptors (BOW and HOG-BOW) on our Wild-Anim dataset. The results in Table 1 show the classification performances obtained from the combination of L2-SVM with local feature descriptors and the results of the deep CNN approaches on our dataset. The results show that the BOW and HOG-BOW methods perform much worse when compared to some scratch and all fine-tuned versions of the deep CNN techniques.

Table 1: Performances of the 16 different techniques on the Wild-Anim dataset

Methods	**Test Accuracy**
OFT-GoogleNet (Top-1)	**99.93**±0.14
OFT-AlexNet (Top-1)	96.80±2.13
OS-GoogleNet (Top-1)	90.00±3.41
OS-AlexNet (Top-1)	82.40±4.92
RFT-GoogleNet (Top-1)	**99.93**±0.14
RFT-AlexNet (Top-1)	97.40±2.15
RS-GoogleNet (Top-1)	89.00±4.05
RS-AlexNet (Top-1)	83.40±5.84
BOW-Color with Max-Pooling	84.00±2.19
BOW-Color with Sum-Pooling	82.40±1.62
BOW-Gray with Max-Pooling	82.00±3.58
BOW-Gray with Sum-Pooling	81.40±2.24
HOG-BOW-Gray with Sum-Pooling	82.60±1.74
HOG-BOW-Gray with Max-Pooling	78.40±1.74
HOG-BOW-Color with Sum-Pooling	73.20±3.37
HOG-BOW-Color with Max-Pooling	63.60±3.01

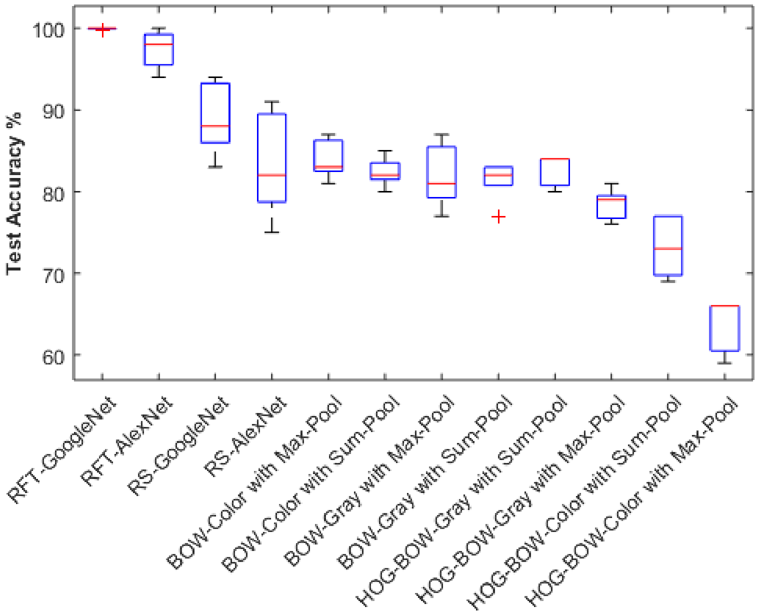


Figure 5: Performance evaluation of our modified versions (RFT and RS) of deep CNN architectures and the BOW variants on 5 test sets. See Table 1 for performances of the original methods (OS and OFT).

The performances on the five different test sets obtained from the proposed deep CNN and the BOW variants applied on our dataset are shown in Figure 5. This figure shows that the results on different test sets are fairly consistent. It also shows the quartile ranges between (Q_1 to Q_3). From the results in Table 1, it can be seen that both RFT-GoogleNet and OFT-GoogleNet outperform every other method with a Top-1 loss rate of 0.07%. Next to it, the best performances are obtained with RFT-AlexNet and OFT-AlexNet with a Top-1 loss rate of 2.6% and 3.2% respectively. These results uncover the high level of performance. Although the ImageNet dataset contains different images of animals as those present in our dataset, the use of having more images and image labels significantly contributes to the outstanding performances of the pre-trained models of AlexNet and GoogleNet. The pre-trained models provide a big advantage to the evaluation of the performances on our dataset. One can therefore argue that the fairest results are from the scratch versions of the GoogleNet architectures which also outperform all the BOW methods.

The scratch versions for both architectures obtain a Top-1 loss rate of 10% for OS-GoogleNet and 11% for RS-GoogleNet, while the results on the scratch AlexNet architecture are much lower. It can be seen from Table 1 that RFT-AlexNet outperforms the OFT-AlexNet by 0.6% and the RS-AlexNet outperforms OS-AlexNet by 1%. However, the OS-GoogleNet outperforms the RS-GoogleNet by 1%. These differences are all not significant, however. We also expected a performance improvement in the final accuracy of the reduced versions, since the training set is not very large. It seems that the used dropout regularization performs very well in this case to prevent overfitting.

The most competitive local descriptor is BOW-color with the max-pooling strategy at 84% which outperforms OS-AlexNet by 1.6%, RS-AlexNet by 0.6% and HOG-BOW-gray with sum-pooling. This may be caused by a rich preservation of color image information from our animal images with the use of BOW-color using max-pooling. However, when we compare the performance of the best BOW variant to the (non-RS-AlexNet) CNN results which start at 89%, its performance is much worse. The second best local descriptor is HOG-BOW-gray with sum-pooling which is better than the other BOW variants.

The worst performing method of our comparison is HOG-BOW-color for both kinds of spatial pooling strategies. HOG-BOW-color obtains the lowest performance with a high computing time between $23 < t \leq 26$ hours compared to CNN methods that use less than $t \leq 2.1$ hours for the overall computation. HOG-BOW-color with both spatial approaches is poor in handling high color resolution feature vectors and requires lots of computing time. From all BOW results, we can see that BOW outperforms the HOG-BOW technique.

Also, the modified CNN architectures are competitive when compared to the original deep CNN architectures but require less computing time. This is evident based on the significant decrease in time by 27% and 26% for both the fine-tuned and scratch versions of the AlexNet architectures. There is no significant improvement in the computing time of the modified version of the GoogleNet architecture compared to the original GoogleNet architecture.

We further carried out an additional performance evaluation using the reduced versions of the Deep CNN on another set of 1000 images from our original dataset. This time all the 1000 images were used as testing set with

10× the images present in the earlier testing set. We ensure that the new testing set is not overlapping with images present in the previous subset that contains 1000 images from our original dataset. This is achieved by performing a fixed split partitioning. In our later experiments, the new testing set is fixed and it is evaluated using 5 different train-validation models generated based on the earlier experimental settings. We computed the mean of 5 runs from our test evaluation, which is reported in Table 2. The results are fairly consistent compared to the earlier results reported in Table 1. This implies that the reduced Deep CNN architectures have an outstanding generalization.

Table 2: Performance evaluation of reduced CNN on another test set

Methods	RFT-GoogleNet	RFT-AlexNet	RS-GoogleNet	RS-AlexNet
Test Accuracy	99.38±0.44	96.72±0.21	89.74±0.85	84.82±1.16

2.6 Discussion

In this chapter, several image recognition techniques were compared on a novel dataset consisting of wild animals. From the results, a conclusion can be drawn that the performance of almost all CNN architectures is much better than the performance of the different bag-of-words techniques. The pre-trained GoogleNet and AlexNet architectures perform exceptionally well, but being trained on ImageNet that contains the same classes, but different images, this does not come as a big surprise. Furthermore as seen from the comparison of the performances of GoogleNet and AlexNet when trained from scratch, the use of GoogleNet performs much better. It is remarkable that the recognition accuracy is still very high even for the relatively small dataset.

Additionally, this research demonstrated that the reduction in the number of neurons in the last inception layer of the GoogleNet and fully connected layers in AlexNet have shown to be competitive when compared to the original GoogleNet and AlexNet architectures. The merit of this approach is that it can significantly decrease the computing power usage. In addition to the contributions to deep learning, we report that the effect of color on BOW with the max-pooling strategy is relatively competitive compared to the AlexNet architecture when trained from scratch. Finally, the BOW technique outperforms the HOG-BOW method.

Future work should involve the application of segmentation and data augmentation techniques on the used dataset. We also want to study the effect of different color spaces using deep learning architectures.

3 ROTATION MATRIX DATA AUGMENTATION

In deep learning, data augmentation is important to increase the amount of training images to obtain higher classification accuracies. Most data-augmentation methods adopt the use of the following techniques: cropping, mirroring, color casting, scaling and rotation for creating additional training images. In this chapter, a novel data-augmentation method that transforms an image into a new image containing multiple rotated copies of the original image in the operational classification stage is proposed. The proposed method creates a grid of $n \times n$ cells, in which each cell contains a different randomly rotated image and introduces a natural background in the newly created image. This algorithm is used for creating new training and testing images, and enhances the amount of information in an image. For the experiments, we created a novel dataset with aerial images of cows and natural scene backgrounds using an unmanned aerial vehicle, resulting in a binary classification problem. To classify the images, we used a convolutional neural network (CNN) architecture and compared two loss functions (Hinge loss and cross-entropy loss). Additionally, we compare the CNN to classical feature-based techniques combined with a k-nearest neighbor classifier or a support vector machine. The results show that the pre-trained CNN with our proposed data-augmentation technique yields significantly higher accuracies than all other approaches.

This chapter was published in:

Okafor, E., Schomaker, L.R.B., and Wiering, M.A. (2018). An Analysis of Rotation-Matrix and Color Constancy Data Augmentation in Classifying Images of Animals. *Journal of Information and Telecommunication, ISSN 2475-1839, Vol 2: 4, pages 465-491.*
Okafor, E., Smit, R., Schomaker, L.R.B., and Wiering, M.A. (2017). Operational data augmentation in classifying single aerial images of animals. *INnovations in Intelligent SysTems and Applications (INISTA), The IEEE International Conference on pages 354-360.*

The use of unmanned aerial vehicles (UAV) has a lot of potential for precision agriculture as well as for livestock monitoring. A previous study (Zhang and Kovacs, 2012) recommended that the combination between precision agriculture and remote sensing and UAV methods can be very beneficial for agricultural purposes. Other research (Katsigiannis et al., 2016; Lukas et al., 2016; López-Granados et al., 2016) has examined this area of research with the use of UAVs for different tasks. A novel area of research is recognizing aerial imagery with the use of deep neural networks. The study in (Lin et al., 2015) demonstrates that the use of a convolutional neural network for ground-to-aerial localization yielded a good performance on some datasets. Another interesting study is the use of deep reinforcement learning for active localization of cows (Caicedo and Lazebnik, 2015). Next to the task of localization, there exists some recent research on the use of UAVs for motion detection and tracking of objects. The study in (Fang et al., 2016) analysed the merits of the use of optical flow with a coarse segmentation approach for aerial motion detection of animals from several videos. Furthermore, in (Gonzalez et al., 2016) the authors extended the idea of using UAVs with object detection and tracking algorithms for monitoring wildlife animals. Another approach is detection and tracking of humans from UAV images using local feature extractors and support vector machines (Imamura et al., 2016).

The idea of data augmentation (DA) has been successfully applied to UAV data as well. In (Jeon et al., 2017), the authors studied augmentation of drone sounds using a publicly available dataset that contains several real-life environmental sounds. Furthermore, the research in (Charalambous and Bharath, 2016) explored the use of a DA method for training a deep learning algorithm for recognizing gaits. Another interesting use of DA is the development of a model for 3D pose estimation using motion capture data (Rogez and Schmid, 2016).

Most of the previous data-augmentation techniques transform a training image to multiple training images using techniques such as: cropping, mirroring, color casting, scaling and rotation. In this chapter, we propose a novel data-augmentation method that transforms a single input image to another image containing $n \times n$ rotated copies of the original image. This method enhances the amount of information in an image, especially if the image contains a single object like in our study (cow or non-cow background). The aim of this chapter is to assess if this novel data-augmentation method leads to higher classification accuracies when combined with different machine learning techniques such as convolutional neural networks or classical feature descriptors on a novel dataset containing aerial images of animals.

Contributions

This chapter proposes a novel data-augmentation technique that transforms a train or test image into a novel single image with multiple randomly rotated copies of the input image. To combine the different rotated images, the proposed method puts them in a grid and adds realistic background pixels to glue them together. This approach presents some merits: 1) It provides more informative images which may aid to yield higher accuracies, 2) It does not require an increase in the number of training images compared to other conventional data-augmentation methods, and 3) The method can also be used to perform data augmentation on test images in the operational stage. The utility of the proposed approach is evaluated by using a CNN which is derived from the original GoogleNet (Szegedy et al., 2015) architecture by keeping only several inception modules. For training this CNN we evaluate if there are differences in using the cross-entropy loss function (softmax classifier) compared to using a Hinge loss function. Furthermore, we compared the CNNs to several classical computer vision techniques using original images and data-augmented images. All techniques were used to investigate the recognition accuracies of aerial images of cows in natural scenes, for which we created our own dataset with an unmanned aerial vehicle. The results show that using fine-tuned CNN models with the proposed data-augmentation technique leads to significantly better results than all other approaches.

Outline. This chapter is organized as follows; Section 3.1 describes the used UAV dataset and the proposed data-augmentation technique. Section 3.2 discusses the methods used for classifying the aerial imagery. The experimental results of the derived CNN and the classical techniques are reported in Section 3.3. Finally, the conclusion is presented in Section 3.4.

3.1 Dataset and Data Augmentation

3.1.1 Dataset Collection

We employed the DJI Phantom 3 Advanced Unmanned Aerial Vehicle (UAV) for collecting video frames of cows and natural backgrounds at different positions and orientations. An illustration of the UAV is shown in Figure 6.

Figure 6: A photo of the UAV used for this study

We applied manual cut-outs with a fixed size of 100 × 100 pixels to obtain positive samples of images that contain a cow, while we employed an automatic extraction of negative samples which have no presence of cows in the image. We flew the drone three times over different fields containing cows in order to obtain different samples. A summary of the three subsets of the obtained images with the amount of positive and negative samples, the video streaming time, and the amount of unique objects is reported in Table 3.

Table 3: Statistics of video records and annotated datasets

	Video ID	Time (s)	Unique Objects	Positive Samples	Negative Samples
1	Subset 1	11	10	37	225
2	Subset 2	43	82	475	2094
3	Subset 3	22	10	50	1100

The unique objects denote cows that are recorded at different time frames and therefore have different appearances in time. Figure 7 shows some samples of images of our aerial dataset.

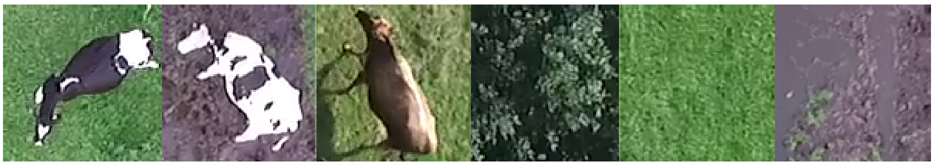


Figure 7: Sample images of the aerial dataset, showing the presence of cow (positive samples) and non-cow (negative samples). Please note that non-cow images are also diverse.

3.1.2 Cross-Set Splits

We used cross-set splits whereby each recorded subset is considered as a separate fold. One subset is used for testing and the other subsets are used for the training set. This process is repeated for the three available subsets. The classical feature descriptors combined with supervised learning algorithms and the derived CNN technique are employed for determining the existence of cows in the natural images. We maintain the same dataset splits for all the experiments using the CNN and the feature extraction techniques. The classical techniques employ two image resolutions; 100 × 100 and 250 × 250 pixels, and for the experiments carried out with the derived CNN we only used 250 × 250 pixels.

3.1.3 Multi-Orientation Data Augmentation

We propose a new offline data-augmentation algorithm called rotation-matrix data-augmentation (ROT-DA) that transforms an input image to a new single image containing multiple randomly rotated versions put in $n \times n$ cells. The use of a larger value for n leads to a new image containing more different poses. The value of n was set to 4 in the experiments, because using higher values of n resulted in making the cow images look very small. An illustration of the proposed data-augmentation method and the overall classification system using the CNN is shown in Figure 8.

The pseudo-code in Algorithm 1 explains the various transformations of the original image to obtain the multi-orientation image.

After inserting the images in the newly created image, background pixels are added to glue them together. This is done by using the nearest-neighbor pixels around the edges of the images. We will also perform experiments with ROT-DA without rotations (ROT-DA-NR), but we do this only for the classical feature-based techniques.

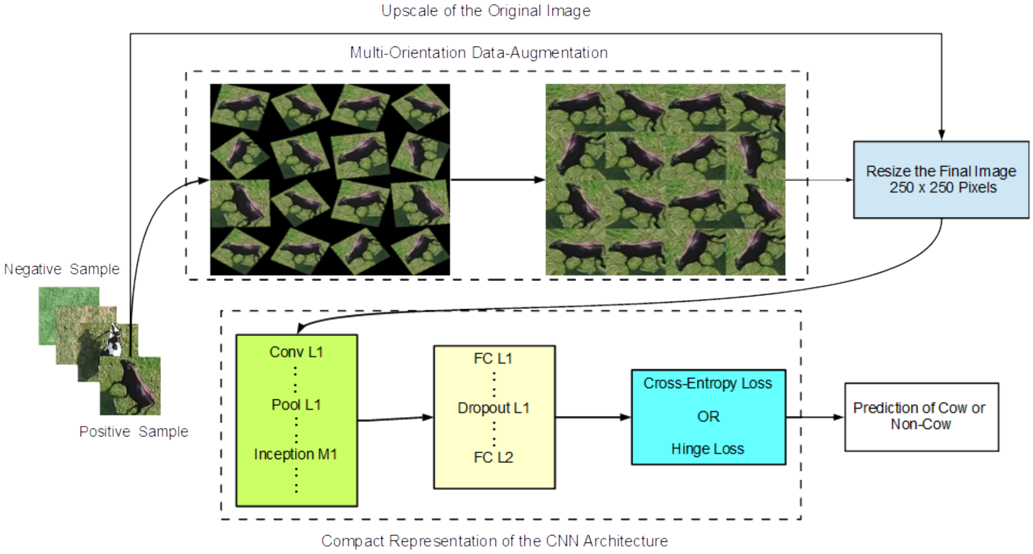

Upscale of the Original Image
Multi-Orientation Data-Augmentation
Resize the Final Image 250 x 250 Pixels
Negative Sample
Positive Sample
Conv L1
Pool L1
Inception M1
FC L1
Dropout L1
FC L2
Cross-Entropy Loss
OR
Hinge Loss
Prediction of Cow or Non-Cow
Compact Representation of the CNN Architecture

Figure 8: Block diagram illustrating the proposed method and overall system using the CNN. The column (':') symbol between different layers represents the connections of neural network layers within the derived CNN architecture. The data-augmented image on the top-left is a multi-orientation image without padding and the image on the top-right is the resulting multi-orientation image with padding.

Algorithm 1 Multi-Orientation Data-Augmentation Algorithm

Input : Given images $I_i(x, y)$ from an input directory, where x, y denote the pixel row and column, and a grid size of $n \times n$.

Output : The data-augmentated versions of the images.

```
procedure Construct a filelist with $N$ images from an input directory.
    for each image $I_i$, $i \in N$ do
        Initialize the total number of cells $n \times n = M$
        for each image $I_i$, for all cells $m \in M$ do
            Define the size of the image resolution.
            Compute a pad-size $I_q = ceil((size(I_i))/2)$.
            Compute a pad-array $I_p$ using a pixel replication padding technique, given $I_i$, $I_q$, pad value set to 'replicate' and the pad direction set to 'both'.
            Rotate $I_p$ with a random angle within the bound $[1^o, 180^o]$, this yields a new image $I_r$.
            Adjust the image $I_r$ to $I_a$ such that undesired background introduced during rotation is filled with artificial pixels from the nearest-neighbor pixels.
            Concatenate each $I_a$ into $M$ cells.
            $I_c = [I_a(\mathbf{k})...I_a(\mathbf{k} + n - 1); ...; ...I_a(M = n^2)]_{n \times n}$ Given that $\mathbf{k} = 1$, $\forall$ $M$ cells, the ellipses (...) denote the column cells entries containing rotated sub-images, and the semicolon (;) in this study represents the start of a new row. Note that each cell in the $n \times n$ grid of cells contains a rotated copy of the input image $I_a(\mathbf{k})$ in a reduced size.
        end for
        Convert the cell structure of $I_c$ into a matrix $I_m$.
        Resize the image $I_m$ to $250 \times 250$ pixels.
        Store each $I_m(i)$ into an output directory
    end for
end procedure
```

3.2 Image Recognition Methods

3.2.1 Three Inception Module CNN Architecture

This architecture is directly derived from the famous GoogleNet architecture as proposed in (Szegedy et al., 2015). We eliminated all the layers after the inception 4a module, except for layers which lead to the first classifier. We do this because the problem under study is more of a binary classification problem and the dataset is quite small. We want to know how the reduced architecture can handle this problem compared to the original GoogleNet. Another modification made with respect to the original GoogleNet architecture is the use of Nesterov's Accelerated Gradient Descent (NAGD) rather than using the conventional stochastic gradient descent (SGD) to update the weights in the deep neural network. The NAGD optimization update rule (Sutskever et al., 2013) is described in equations 20 and 21:

$$u_{i+1} = \mu u_i - \alpha_L \nabla L \left(W_i + \mu u_i \right) \tag{20}$$

$$W_{i+1} = W_i + u_{i+1} \tag{21}$$

where $L \in \{L^h, L^c\}$ is the loss function, μ is the momentum value, α_L is the learning rate, u_i is the momentum variable, ∇ is the rate of change in L, i is the iteration number and W_i denote the learnable weights. We employed randomly initalized weights for the scratch CNN and pretrained weights from the ImageNet dataset for the fine-tuned CNN (GoogleNet architecture). In addition to our modification, we remark that the original GoogleNet (in Caffe framework) uses a simple online data-augmentation that involves cropping (with a default crop size of 224 × 224 pixels), i.e. cutting out several patches from an input image at 5 positions (as five in a dice), and additionally flipping (horizontal reflection) to obtain more samples. During training of the CNN model, it automatically flips each cropped image to double the effective dataset size. The cropping means an act of extracting some portions from an input image. In our customized CNNs, we considered the original and two additional crop sizes: 125 × 125 and 250 × 250 pixels. The crop size of 250 × 250 implies the single actual size of the input image. Furthermore, we evaluated flip and non-flip conditions. All the input images to the CNN have image sizes of 250 × 250 pixels. For the ROT-DA image, each cell of the 4 × 4 grid contains a copy of the input image in a reduced size and the method fills up empty spaces with nearest neighbor pixels.

Table 4: Three inception module convolutional neural network architecture

Layer Type	Patch Size / Stride	Output Size	Depth	Number of Convolutional Filters	Blob Parameters
Conv 1	$7 \times 7/2$	$112 \times 112 \times 64$	1		16.06M
Max Pool 1	$3 \times 3/2$	$56 \times 56 \times 64$	0		4.01M
Conv 2	$3 \times 3/2$	$56 \times 56 \times 192$	2	0, 64, 192, 0, 0, 0	12.04M
Max Pool 2	$3 \times 3/2$	$28 \times 28 \times 192$	0		3.01M
Inception 3a		$28 \times 28 \times 256$	2	64, 96, 128, 16, 32, 32	4.01M
Inception 3b		$28 \times 28 \times 480$	2	128, 124, 192, 32, 96, 64	7.53M
Max Pool 3	$3 \times 3/2$	$14 \times 14 \times 480$	0		1.88M
Inception 4a		$14 \times 14 \times 512$	2	192, 96, 208, 16, 48, 64	2.01M
Average Pool 1		$4 \times 4 \times 512$	0		163.84K
Top Conv-1	$1 \times 1/1$	$4 \times 4 \times 128$	1		40.96K
FC 1 / 70% Dropout layer		$1 \times 1 \times 1024$	1 / 0		20.48K
FC 2		$1 \times 1 \times 2$	1		0.04K
Cross Entropy (Softmax) / Hinge Loss		$1 \times 1 \times 2$	0		

The derived three inception module CNN architecture is described in Table 4. This architecture involves the use of three inception modules that allow the concatenation of filters of different dimensions and sizes into a single new filter (Shin et al., 2016). In each inception module, there exist six convolution layers and one pooling layer. Moreover, there exist several rectifiers (ReLUs) which are placed immediately after the convolutional and fully-connected layers. Furthermore, there exist four pooling layers excluding those within the inception modules, two bottom convolutional layers and one top convolutional layer which comes after the average pooling layer. We use one top-1 loss function which employs either the Hinge loss or the cross-entropy loss (for the Softmax classifier). The L_1-norm Hinge loss L^h used in our study can be defined as:

$$L^h(x_i) = \frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{K} \left(\max \left(0, 1 - y_i^k z_k(x_i) \right) \right) \tag{22}$$

where $y_i^k = \{1, -1\}$, $y_i^k = 1$ if x_i belongs to the target class of the k-th class output unit, and $y_i^k = -1$ if x_i does not belong to the target class. The variable N denotes the total number of training images in a batch. K accounts for the number of class labels and $z_k = x^T w$ is the final

activation of the output units. Here, $x \in \mathbb{R}^D$ denote the D-dimensional features of the previous hidden layer, and the learnable weights of the last layer are $w \in \mathbb{R}^{D \times K}$.
The Cross-Entropy Loss L^c used in our study is defined as:

$$L^c(x_i) = -\frac{1}{N} \sum_{i=1}^{N} y_i \left(log \left(\frac{exp(z_i(x_i))}{\sum_{k=1}^{K} exp(z_k(x_i))} \right) \right) \tag{23}$$

where the y_i denote the target values $y_i \in \{0, 1\}$. The fraction within the log accounts for the softmax activation function (Okafor et al., 2016), which computes the probability distribution of the classes in a multi-class classification problem. Note that in this study we are dealing with a binary classification problem and we use 2 output units in the CNN.

The CNN under study consists of two fully connected (FC) layers: FC 1 with a corresponding ReLU computes the hidden unit activations, which is immediately followed by a regularization dropout of 0.7, and the second FC 2 contains the output neurons which represent the negative and positive class. The working operations of the CNN are well explained in the paper (Szegedy et al., 2015).

3.2.1.1 *CNN Experimental Setup*

All experiments were run on the Caffe deep learning framework on a Ge-Force GTX 960 GPU model. The used experimental parameters are as follows: training display interval is set to 40, average loss is set to 40, learning rate is set to 0.001, learning policy is set to step, the step size is set to 4000 iterations, power is set to 0.5, gamma is set to 0.1, the momentum value is set to 0.9, weight decay is set to 0.0002, and maximum iteration is set to 10000, which generates a snapshot model after every 500 iterations (which represent a snapshot). This resulted in 20 snapshots for the entire training process. The mentioned parameters were not altered during all the experiments for the different model configurations. The training images from the combination of any of the two subsets as reported in Table 3, is further split into the ratio 80% for training and 20% for validation. We employed a training batch size set to 20 and testing batch size set to 5 for all experiments, but with different test iterations. The altered parameters for the three subsets of the aerial dataset used with their corresponding splits are described in Table 5.

Table 5: CNN parameters and dataset split information

Parameters	**Subset 1**	**Subset 2**	**Subset 3**
Test Images	262($\sim$ 7%)	2569($\sim$ 65%)	1150($\sim$ 29%)
Training Images	2975($\sim$ 74%)	1129($\sim$ 28%)	2264($\sim$ 57%)
Validation Images	744($\sim$ 19%)	283($\sim$ 7%)	567($\sim$ 14%)
Total Images	3981(100%)	3981(100%)	3981(100%)
Solver Test Iteration (Val/Train)	148	56	113
Test Iterations for Evaluation	52	514	230

We first performed experiments with both the original and our derived CNN trained from scratch on the original images. The preliminary results show that our proposed architecture requires less memory usage and a decrease in training computing time. This is summarized in Table 6. Additionally, our architecture obtains a similar level of performance compared to the original CNN.

Table 6: Preliminary experiment using original and our proposed CNN on three cross splits of the aerial image dataset

Evaluation/Methods	**Derived CNN, NAGD**	**Original CNN, NAGD**
Time (min)	$25.1 \leq t \leq 26.8$	$63.2 \leq t \leq 69.1$
Memory Usage (MB)	752	1079
Average Validation %	99.94	99.94
Average Test %	97.87	97.71
Time Improvement %	61.3 (decrease)	-

3.2.2 Classical Features Combined with Supervised Learning Algorithms

In this subsection, we describe the three feature extraction techniques which we use and combine with the k-nearest neighbor classifier and the support vector machine (SVM) with a linear kernel or a radial basis function (RBF) kernel.

3.2.2.1 *Color Histogram*

The color histogram (Color-Hist) is a feature extraction technique that analyses the pixel color values within an image. For this, the pixel color values of an image which exist as RGB (Red, Green and Blue) are first transformed to HSV (Hue, Saturation, and Value). After that, the value of each pixel in a channel is put in a histogram consisting of different bins. In the experiments, only the saturation channel with a bin size of 32 is used, because it obtained the best performance in preliminary experiments. The resulting feature vector containing 32 values is given to the supervised learning algorithms.

3.2.2.2 *Histogram of Oriented Gradients*

The histogram of oriented gradients (HOG) (Dalal and Triggs, 2005) feature descriptor analyses patches (local regions) from an image. Then histograms are constructed based on the occurrences of orientation gradients within the patches. The HOG descriptor can process gray or color image information. In this study, we only considered the gray option. The procedure for constructing the HOG is as follows: convert the color images of the aerial imagery into grayscale, then compute the gradients with two gradient kernels to compute the gradient values for each pixel from the grayscale image. The gradients for each pixel within a small block (cell) are put in bins (Junior et al., 2009; Takahashi et al., 2014), where each bin defines a specific orientation range. The following parameters were used, because they worked best in preliminary experiments: a grid of 2×2 blocks is used, where each block is split into 2×2 cells. The number of orientation bins is set to 4. This results in a feature dimension size of 64. This feature vector is fed as input to the supervised learning algorithms.

3.2.2.3 *The Combination of HOG and the Color Histogram*

In this technique, the features from both the HOG and Color-Hist are combined to form the HOG-Color-Hist feature descriptor. The features from both HOG and Color-Hist are first computed separately. The optimal parameters used for HOG in the combined feature are different from the HOG descriptor alone, because they gave slightly better results in the preliminary experiments. The HOG parameters used in this technique use 32×32 pixels per cell, for which we used 9 cells in total from 100×100 pixel images with a single block. The number of orientation bins is set to 4 and the final feature dimensionality is 36. We used the hue channel from the color-histogram with 32 bins. These features are normalized and concatenated to obtain the final feature vector with 68 elements.

Several experiments were conducted to determine the best choice of parameters for the used classifiers with the different classical feature descriptors. For the K parameter in K-nearest neighbor (KNN) we tried $K = \{1, 2, 3, 4, 5, 10\}$. The C parameter of the linear SVM is set to $C = 2^{q-1}$, with the explored values $q \in \{1, 2, ..., 19\}$. For the SVM with the RBF kernel, we tried $C = \{1, 2, 3, 5\}$ with $\gamma = 10^{p-1}$, where $p \in \{1, 2, ..., 4\}$. The optimal parameters used for each of the classifiers are reported in Table 7. All the algorithms used for the classical techniques were developed in Python.

Table 7: Best found parameters used for the various classifiers with the classical feature descriptors

Classical Techniques	**RBF-SVM**	**Linear SVM**	**K-NN**
HOG	$C = 3$, $\gamma = 1000$	$C = 8$	$K = 1$
Color-Hist	$C = 1$, $\gamma = 100$	$C = 8192$	$K = 3$
HOG-Color-Hist	$C = 1$, $\gamma = 100$	$C = 256$	$K = 3$

3.3 Results

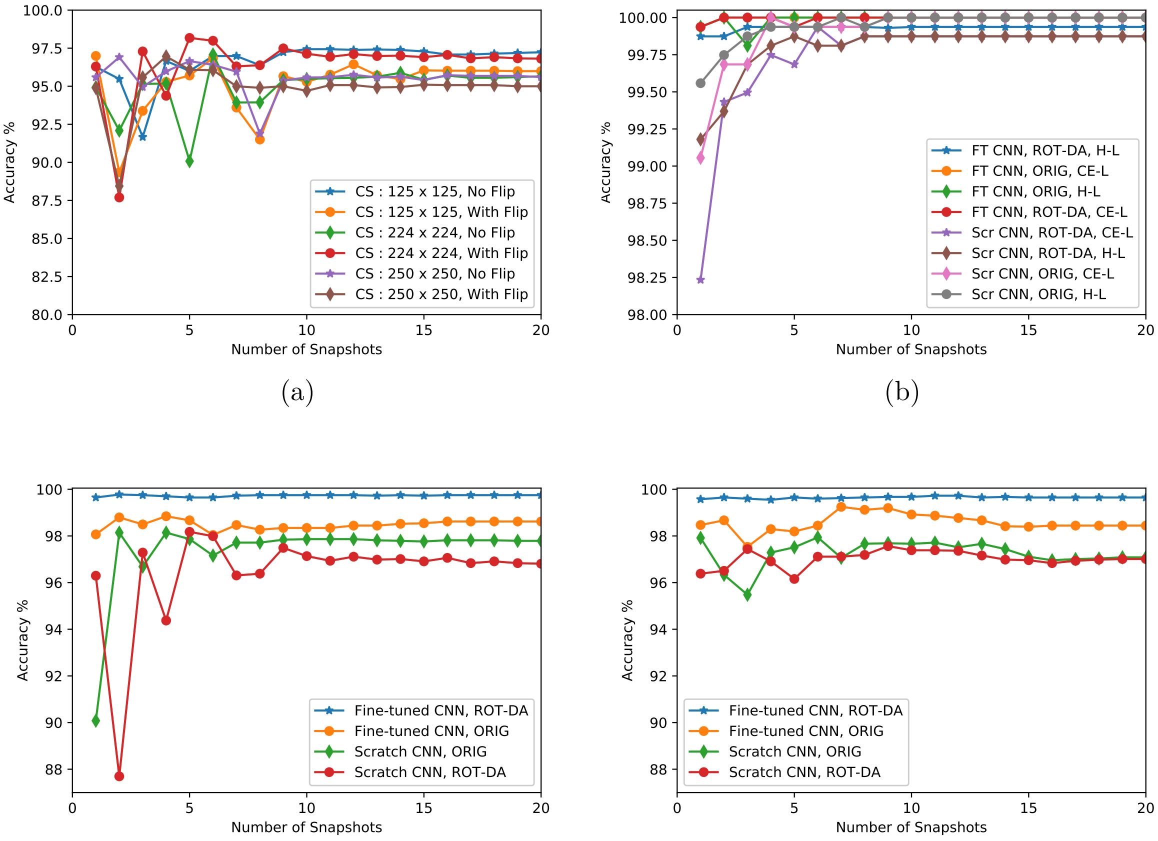
To compute the average results of the different subsets of this dataset, we compute the weighted average accuracy, which is computed by summing over the relative testing dataset sizes multiplied with the average accuracies on the testing datasets. The weighted mean can be computed using the expression: $T_m = \frac{\sum_{s=1}^{S} W_s T_s}{\sum_{s=1}^{S} W_s}$, where T_m denotes the weighted mean test accuracies, W_s denote the weights, which represent the number of individual images per test subset $W_s = \{262, 2569, 1150\}$, and T_s are the test accuracies for the various subsets, with $S = 3$.

3.3.1 Evaluation of the CNN Architecture

In our preliminary studies, we carried out experiments on the data-augmentation (ROT-DA) version of our dataset to determine the optimal crop size. We used models generated from the train-validation experiments for evaluating our test sets. We initially employed the scratch CNN with the cross-entropy classification loss, which is combined with or without flipping and with different crop sizes: 125×125, 224×224, and 250×250. The results of these experiments are shown in Figure 9a, and suggest that the optimal method uses a crop size of 224×224 pixels with flipping. This yields an accuracy of 98.18% that occurred at the 5^{th} snapshot. We observed in general that there exist marginal differences between the various settings.

Based on this outcome, we used the best crop size with flip settings to carry out the experiments using the scratch and fine-tuned versions of the CNN. For this, we used both the data-augmented dataset (ROT-DA) and the original (ORIG) images. The validation results from Figure 9b show that the scratch and the fine-tuned CNN applied on the two kinds of images converge to a near maximum level of performance. The reason for this lies in the fact that most of the validation images contain similar objects as in the training set. The validation results at the 5^{th} snapshot are reported in Table 8. From the table, we can see that the use of the original dataset leads to more overfitting. The results of the different CNNs with the cross-entropy loss function are shown in Figure 9c. From this figure we can observe that the best obtained test accuracy is obtained by the fine-tuned CNN applied on the ROT-DA images in the 2^{nd} snapshot. We further investigated the CNN with the L_1 Hinge Loss, using the earlier mentioned CNN settings (scratch and fine-tuned versions) applied on the two sets of images (ROT-DA and ORIG). The results obtained are shown in Figure 9d.

Based on the performances recorded during this preliminary investigation, we only compared results obtained at the 5^{th} snapshot as reported in Table 8. The results show that the fine-tuned CNN trained on the data-augmented images yields higher test classification accuracies when compared to the fine-tuned CNN trained on the original images of the dataset. We compared the different approaches using the binomial distribution of correctly classifying test images. The results show that the



(a)

(b)

(c)

(d)

Figure 9: Weighted mean classification accuracy on the Aerial UAV Dataset while training for 10,000 iterations (20 snapshots); where (a) Preliminary test performance using scratch CNN with cross-entropy loss (softmax classifier) applied on ROT-DA alone using different crop sizes (CS), with and without flips. (b) Validation set evaluation of the CNN with cross entropy loss (CE-L) and Hinge Loss (H-L) using a crop size of 224×224 and flip. The ROT-DA means the augmented dataset and ORIG means the originally up-scaled images. The FT means Fine-tuned and Scr means Scratch. (c) Test evaluation of the CNN with CE loss using a crop size of 224×224 and flip, and (d) Test evaluation of the CNN with L1-Norm Hinge Loss using a crop size of 224×224 and flip.

fine-tuned CNN trained on the data-augmented images yields significantly higher classification accuracies ($p < 0.01$) when compared to the fine-tuned CNN trained on the original images of the dataset. Overall, the fine-tuned CNNs obtain the best results and combined with the data-augmented images, the results are very good (99.65%). Finally, the results show that overall the use of the cross-entropy loss function leads to better results than the use of the Hinge loss function.

Table 8: Weighted mean of the test and validation classification accuracies of the CNN applied on the aerial imagery dataset after 5 snapshots

Evaluation	Method	Cross Entropy Loss	Hinge Loss
Test	Fine-tuned CNN, ROT-DA	99.65	99.65
	Fine-tuned CNN, ORIG	98.67	98.19
	Scratch CNN, ROT-DA	98.18	96.16
	Scratch CNN, ORIG	97.87	97.51
Validation	Fine-tuned CNN, ROT-DA	99.94	99.94
	Fine-tuned CNN, ORIG	100.00	100.00
	Scratch CNN, ROT-DA	99.68	99.81
	Scratch CNN, ORIG	99.94	99.94

3.3.2 Evaluation of Classical Descriptors

The weighted mean test accuracies of the classical techniques on the aerial imagery dataset are reported in Table 9. We observe that the RBF-SVM outperforms the other two classifiers (K-NN and linear SVM) when combined with each of the feature descriptors. Another observation is that the classifiers with the Color-Hist or HOG-Color-Hist features yield better performances than using the HOG descriptor alone. This shows the importance of using color information for this classification problem. Still, the results are significantly worse than the results using the CNN methods.

Table 9 also shows the results of using the RBF-SVM with different datasets and different feature descriptors using larger images (250×250 pixels). The results show that here data-augmentation does not lead to significantly better results. This can be explained by the fact that the best feature descriptor, the color histogram, is not affected by this data-augmentation method. Finally, we note that the original image with the smaller 100×100 resolution works better for the HOG feature descriptor and therefore also for HOG combined with the color histogram. This can be explained by the fact that we optimized the HOG parameters using the smaller images.

Although the performances of the CNN techniques are much better, the classical techniques have a lower training computing time: $t \leq 1$ min. This is because of the low dimensionality of the extracted features and the low number of trainable parameters.

Table 9: Summary of the weighted mean test performances for all CNNs and classical methods on our dataset. Note that each of the subsets (Sub 1, Sub 2, or Sub 3) represents the test results.

Methods	**Sub 1**	**Sub 2**	**Sub 3**	**Weighted Mean**
Fine-tuned-CNN, ROT-DA, Cross Entropy Loss	100.00	99.73	99.39	99.65
Fine-tuned-CNN, ROT-DA, Hinge Loss	99.62	99.77	99.39	99.65
Fine-tuned-CNN, ORIG, Cross Entropy Loss	99.62	98.29	99.30	98.67
Fine-tuned-CNN, ORIG, Hinge Loss	99.62	97.55	99.30	98.19
Scratch-CNN, ROT-DA, Cross Entropy Loss	98.23	98.72	96.96	98.18
Scratch-CNN, ROT-DA, Hinge Loss	98.08	96.19	95.65	96.16
Scratch-CNN, ORIG, Cross Entropy Loss	98.85	99.34	94.35	97.87
Scratch-CNN, ORIG, Hinge Loss	97.69	98.83	94.52	97.51
RBF-SVM-HOG, ORIG-100×100	96.56	86.99	95.30	90.02
RBF-SVM-Color-Hist, ORIG-100×100	96.56	96.07	96.87	96.33
RBF-SVM-HOG-Color-Hist, ORIG-100×100	96.56	96.11	96.69	96.31
Linear-SVM-HOG, ORIG-100×100	85.88	81.51	95.65	85.88
Linear-SVM-Color-Hist, ORIG-100×100	96.95	93.77	95.83	94.57
Linear-SVM-HOG-Color-Hist, ORIG-100×100	95.80	94.08	93.74	94.09
KNN-HOG, ORIG-100×100	88.17	84.35	96.78	88.19
KNN-Color-Hist, ORIG-100×100	96.56	96.50	94.86	96.03
KNN-HOG-Color-Hist, ORIG-100×100	96.95	96.46	94.78	96.01
RBF-SVM-HOG, ORIG-250×250	85.88	81.51	95.65	85.88
RBF-SVM-Color-Hist, ORIG-250×250	96.57	95.37	96.52	95.78
RBF-SVM-HOG-Color-Hist, ORIG-250×250	85.88	81.51	95.65	85.88
RBF-SVM-HOG, ROT-DA-250×250	85.88	81.51	95.65	85.88
RBF-SVM-Color-Hist, ROT-DA-250×250	96.18	95.25	96.70	95.73
RBF-SVM-HOG-Color-Hist, ROT-DA-250×250	95.04	93.97	96.08	94.65
RBF-SVM-HOG-ROT-DA-NR-250×250	94.66	81.51	86.61	83.84
RBF-SVM-Color-Hist-ROT-DA-NR-250×250	96.56	95.13	96.43	95.60
RBF-SVM-HOG-Color-Hist-ROT-DA-NR-250×250	95.04	91.98	96.43	93.47

3.4 Remarks

We developed a novel data-augmentation method that transforms an image into a new image containing multiple random transformations of the image. The new augmentation method does not lead to an increase in the number of training images compared to previously used data-augmentation techniques. We evaluated this method with deep neural networks and feature descriptors combined with supervised learning algorithms on a new dataset of aerial images of cows.

Our study shows that the use of the data-augmented images leads to the best performances when combined with fine-tuned CNNs. Furthermore, the results show that all CNN approaches significantly outperform the classical approaches with or without the use of data augmentation. The performances of the scratch CNNs are worse than the accuracies of the fine-tuned CNNs with data-augmented images which obtain an accuracy of 99.65%. Furthermore, the RBF-SVM yields better classification performances than the K-NN and a linear SVM when combined with the used feature descriptors. It should be noted that our DA algorithm is useful for the CNNs, because although the used CNNs are more or less translational invariant, they are not rotational invariant.

The idea of our data-augmentation method can be extended by including different techniques to create new images such as color casting with different illumination effects. Furthermore, the proposed data-augmentation technique can also be combined with other data-augmentation methods to create more training images, which may be useful when dealing with small datasets.

4 UNIFICATION OF ROTATION MATRIX AND COLOR CONSTANCY

This chapter addresses the enhancement of images and the problem of over-fitting when training CNN on image classification. We investigate the use of deep learning to assess the classification performance of the rotation-matrix data augmentation combined with color constancy versions of the datasets. For the color constancy methods, we use two well-known retinex techniques: the multi-scale retinex and the multi-scale retinex with color restoration for enhancing both original (ORIG) and rotation-matrix (ROT) images. We perform experiments on three datasets containing images of animals, from which the first dataset is collected by us and contains aerial images of cows or non-cow backgrounds. To classify the images in the three dataset, we use a convolutional neural network (CNN) architecture with cross-entropy loss. This method is used to examine the color-constancy-DA variants, ORIG and ROT-DA alone for three datasets (Aerial UAV, Bird-600 and Croatia fish). The results show that the rotation-matrix DA is very helpful for the Aerial UAV dataset. Furthermore, the color-constancy DA is helpful for the Bird-600 dataset. Finally, the results show that the finetuned CNNs significantly outperform the CNNs trained from scratch on the Croatia fish and the Bird-600 datasets, and obtain very high accuracies on the Aerial UAV and the Bird-600 datasets.

This chapter was published in:

Okafor, E., Schomaker, L.R.B., and Wiering, M.A. (2018). An Analysis of Rotation-Matrix and Color Constancy Data Augmentation in Classifying Images of Animals. *Journal of Information and Telecommunication, ISSN 2475-1839, Vol 2: 4, pages 465-491.*

Data augmentation has often been used in deep learning to increase the number of training images to obtain high classification accuracies. Previous approaches to data augmentation use cropping, rotation, illumination, scaling, and color casting for creating more training images. A recent research by Pawara et al. (2017) examined the classification performances of two CNN methods (AlexNet and GoogleNet) with several data-augmentation techniques for different plant datasets. This research investigates the rotation-matrix and color-constancy algorithms as methods for data-augmentation with the objective to use one or more machine learning algorithms to classify images within three animal datasets.

Some researches have considered rotating plant images in different angular positions while the effect of white or zero pixel values introduced during rotation of the images were not discussed (Pawara et al., 2017; Ghazi et al., 2017), however, their research show that data-augmentation techniques can be used to reduce overfitting and improve the overall performance of the CNN models. Additionally the research by Sladojevic et al. (2016) attempts to develop a plant disease recognition CNN model with three image transformation techniques: affine, perspective, and rotation.

In contrast to the rotation technique as mentioned above, the idea of color constancy algorithms has widely been studied in image processing and computer vision as a method for enhancing the quality of an image while preserving the color information of an object under varying illumination conditions. The research works by Rahman et al. (1996); Jobson et al. (1997) have proposed a multi-scale retinex (MSR) method, which has the prowess to achieve excellent color rendition and dynamic range compression as opposed to their previous works on the single scale retinex (SSR). An improvement was made in the MSR by the authors in (Rahman et al., 2004), who incorporated color restoration to produce a multi-scale retinex for color restoration (MSRCR). Several improvements have been made on MSR to produce variants of the MSR algorithm. One of such methods is the combination of MSR with chromaticity preservation (Petro et al., 2014). Another modification on the MSR is the incorporation of the Autolevel algorithm that removes outliers, and improves the contrast level within an image, and shows computational improvements when used with a graphical processing unit (Jiang et al., 2015).

However, the unification of color constancy and rotation matrix algorithms as a method of data augmentation has received limited attention. This chapter extends the research in (Okafor et al., 2017) by considering the proposed $n \times n$ rotation algorithm together with color-constancy techniques as methods of data augmentation. The proposed techniques are examined on two animal datasets (Croatia fish (Jaeger et al., 2015) and Bird-600 (Lazebnik et al., 2005)) and an aerial image dataset collected using an unmanned aerial vehicle (UAV) (Okafor et al., 2017).

Most of the previous data-augmentation techniques transform a training image to multiple training images using techniques such as: cropping, contrast, illumination, mirroring, color casting, scaling and rotation. In this chapter, we extend the data-augmentation method proposed in (Okafor et al., 2017) that transforms a single input image to another image containing $n \times n$ rotated copies of the original image. This method enhances the amount of information in an image. Additionally, this chapter investigates the use of two well-known color-constancy methods (MSR and MSRCR) for creating more samples of both original and rotation-matrix versions of three datasets: Aerial UAV (Okafor et al., 2017), Croatia Fish (Jaeger et al., 2015), and Bird-600 (Lazebnik et al., 2005). The objective of this chapter is to use convolutional neural networks to assess the classification performance on several variants of the used datasets.

Contributions

This chapter investigates the use of well-known color-constancy techniques (MSR and MSRCR) for creating new image samples of both original (ORIG) and the new rotation-matrix (ROT) images on three datasets: UAV aerial images, Croatia Fish (Jaeger et al., 2015), and Bird-600 (Lazebnik et al., 2005), with the aim to increase the amount of training image samples. This approach enhances the color information of the images which could be very useful to get higher classification accuracies with the CNN. We train the CNN with the cross-entropy loss function and compare the classification performances of the color-constancy-DA (with ORIG/ROT), ORIG alone, and ROT-DA alone on three datasets. The study also considers two broad forms of DA based on their increase (color-constancy-DA) or no increase (ROT-DA alone) in the amount of training images.

The results show that the finetuned CNN with an appropriate selection of the grid resolution and angular bounds for the rotation algorithm combined with color-constancy methods yields the highest classification accuracies on most of the used datasets. Moreover, the results show that using finetuned CNN models with the proposed data-augmentation (ROT-DA) technique on the Aerial UAV images leads to significantly better results than all other approaches. Finally, the results of our proposed approaches to data augmentation combined with the fine-tuned CNN significantly surpass previous results on the Bird-600 dataset (Lazebnik et al., 2005).

Outline. This chapter is organized as follows; Section 4.1 describes the used datasets and the proposed data-augmentation techniques. Section 4.2 discusses the methods used for classifying the Aerial UAV dataset and two other animal datasets. Section 4.3 describes the CNN experimental setups and the results obtained from the various classification methods on the used datasets. Finally, the conclusion is presented in Section 4.4

4.1 Dataset and Data Augmentation

This section provides the descriptions of the used datasets and the data-augmentation techniques.

4.1.1 Datasets

4.1.1.1 *Aerial UAV Dataset*

The description of the Aerial UAV dataset is previously explained in section 3.1.1.

4.1.1.2 *Croatian Fish Dataset*

This dataset was originally presented in (Jaeger et al., 2015). It contains a total of 794 images and has 12 classes with a non-uniform distribution of the images per class. The authors reported an accuracy of 66.78% in their study using a CNN combined with a linear SVM classifier. We adopted a different split in our experiment because of the imbalance of the image samples within the various classes. We ensured that approximately half of the image samples were kept aside as test sets. Figure 10 shows sample images of this dataset for each of the classes.

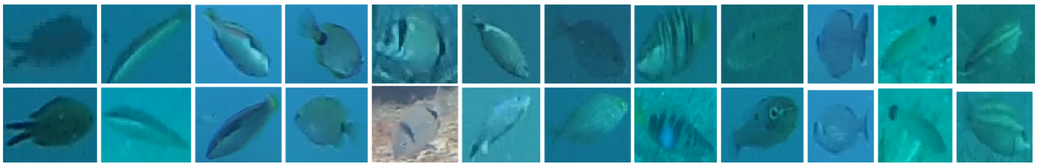


Figure 10: Sample images of the Croatia fish dataset showing each of the fish species (each column): Chromis_chromis, Coris_julis_female, Coris_julis_male, Diplodus_annularis, Diplodus_vulgaris, Oblada_melanura, Sarpa_salpa, Serranus_scriba, Spicara_maena, Spondyliosoma_cantharus, Symphodus_melanocercus, and Symphodus_tinca (Jaeger et al., 2015).

4.1.1.3 *Bird-600 Dataset*

This dataset was originally presented in (Lazebnik et al., 2005). The dataset contains a total of 600 images and has six classes with 100 individual image samples per class. We adopted a similar dataset distribution by keeping 50% of the total image samples as test set as reported in (Lazebnik et al., 2005) in our experiments. The authors reported an accuracy of 92.33% in their study by using a probabilistic part-based method for texture and object recognition. Figure 11 shows sample images of this dataset for each of the classes.

4.1.2 Data Augmentation Techniques

4.1.2.1 *Multi-Orientation Data Augmentation*

We propose a new offline data-augmentation algorithm called ROT-DA that transforms an input image to a new single image containing multiple randomly rotated versions put in $n \times n$ cells. The use of a larger value for n leads to a new image containing more different poses. For the Aerial UAV dataset, the value of n was set to 4 in the experiments, because using higher values of n resulted in making the cow images look very

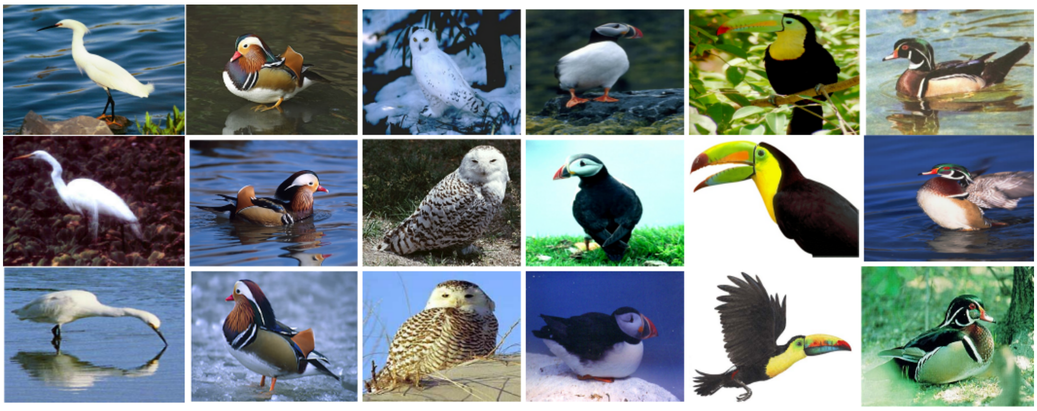


Figure 11: Sample images of the Bird-600 dataset for each of the bird species (each column): egret, mandarin, owl, puffin, toucan and wood_duck (Lazebnik et al., 2005).

small. On the other two animal datasets, we set $n = \{2, 4\}$ for Croatia fish while for the Bird-600, we set the value $n = \{1, 2\}$. An illustration of the proposed data-augmentation method and the overall classification system using the CNN is shown in Figure 8. The pseudo-code in Algorithm 1 explains the various transformations of the original image to obtain the multi-orientation image.

4.1.2.2 *Color Constancy Data Augmentation*

Color constancy is the perception of an object which ensures that perceived colors of objects remain relatively constant under various variations in illumination conditions. This area of study has found relevance in image processing and computer vision. Color constancy uses contrast/lightness enhancement and color rendition for improving the quality of an image. Most color-constancy algorithms use the retinex theory. The idea of the retinex theory was proposed initially by (Land and McCann, 1971). The research by Provenzi et al. (2005) provided the basis for understanding the retinex algorithm from a mathematical standpoint. Our study examines two kinds of multiscale retinex algorithms.

1. Multi-Scale Retinex (MSR): This algorithm was proposed by (Rahman et al., 1996, 2004). The algorithm provides a trade-off between

color rendition and local dynamic range (Petro et al., 2014). MSR computes the weighted sum of the outputs from various single scale retinex (SSR). According to Jobson et al. (1997), an MSR image can be computed as:

$$f_{msr_k}(x,y) = \sum_{m=1}^{\mathbf{M}} W_m f_{m_k}(x,y) \tag{24}$$

$$f_{m_k}(x,y) = log(I_k(x,y)) - log\left(\sum_{(x,y)} C_m exp\left[\frac{-(x^2+y^2)}{2\sigma_m^2}\right] I_k(x,y)\right) \tag{25}$$

where f_{m_k} is the single scale retinex output for $\mathbf{M}$ scales, W_m denote the weights for each scale variable, $W_m = \frac{1}{3}$, the maximum number of scales is $\mathbf{M} = 3$ because the number of the RGB image channels is equal to the number of scales, C_m represents the normalization factor, and $I_k(x,y)$ denotes the image pixel coordinates for a given color band k. The $\sigma_m \in \{15, 80, 250\}$ are the standard deviations of the Gaussians for each of the scales. We adopted the same parameters as used by Jobson et al. (1997); Petro et al. (2014), because they also perform well in our study. Furthermore, we further computed the $f_{msr_k}(x,y)$ by using the mathematical expression proposed by Moore et al. (1991), where each color channel is modified by the absolute minimum and maximum of the RGB color channels. This can be computed as:

$$f_{msr_k}(x,y) = 255\frac{f_{msr_k}(x,y) - min_k(min_{(x,y)} f_{msr_k}(x,y))}{max_k(max_{(x,y)} f_{msr_k}(x,y)) - min_k(min_{(x,y)} f_{msr_k}(x,y))} \tag{26}$$

2. Multi-Scale Retinex with Color Restoration (MSRCR): (Jobson et al., 1997; Rahman et al., 2004) initially proposed the MSRCR algorithm. An MSRCR image f_{msrcr_k} can be computed by the product of color

restoration functions C_k of the chromaticity and the MSR outputs. The modified version of the MSRCR $f_{msrcr_k}(x, y)$ from the research by Petro et al. (2014), can be computed as:

$$f_{msrcr_k}(x, y) = \lambda\left(C_k(x, y) f_{msr_k}(x, y) + \beta\right) \tag{27}$$

$$C_k(x, y) = log\left[\alpha\left(\frac{I_k(x, y)}{\sum_{k=1}^{\mathbf{K}} I_k(x, y)}\right)\right] \tag{28}$$

where α controls the strength of the non-linearity and λ is a constant. For the MSRCR experiment α is set to 125 while λ is set to 0.8 and $\mathbf{K}$ represent the total number of spectral bands $(\mathbf{K} = \mathbf{3})$ while β is set to 46.

Proposed Color-Constancy Data Augmentation: This study examines the possibility of using the original or rotation-matrix images that are fed as input to the MSR or MSRCR algorithm. This process can also be done vice-versa by creating the color-constancy images and then pass them as inputs to the rotation-matrix algorithm. The new images are then combined with either original or rotation-matrix images to obtain either double or three times the effective size of the initial train-validation image dataset. Please note by three times, we mean combining ORIG+MSRCR-ORIG+MSR-ORIG or ROT+MSRCR-ROT+MSR-ROT. We carried out experiments using two animal datasets and the UAV dataset. Some samples of both the original and rotation-matrix images with and without color constancy are shown in Figure 12, Figure 13, and Figure 14 for the Aerial UAV dataset, Croatia fish dataset, and Bird-600 dataset respectively.

We carried out some considerations to the rotational bounds for the ROT-DA alone or color constancy DA with ROT images on the three datasets.

a) For the Aerial UAV and Croatia Fish datasets irrespective of the order of the grid cells, we used a rotational angle in the range $[1^o, 180^o]$.

b) For the Bird-600 experiments, we considered two rotational conditions for 2×2-ROT-DA which we defined in two versions;

   i. Version 1 (V1): the rotational angles for different image poses lie in the bound $[1^o, 180^o]$. This computation was carried out on 2×2-ROT-DA alone and color-constancy DA with 2×2-ROT images separately.

   ii. Version 2 (V2): the rotational angles for different image poses lie in the bound $[-15^o, 15^o]$ and we exempted angle 0^o in our computation; this is because we do not want to have the existence of the original image twice in the new DA variants. This computation was carried out only on the color constancy DA with 2×2-ROT images.

c) On the Bird-600 experiments, we also considered the color constancy DA with 1×1-ROT which used the same angular rotation bounds as in V2. This setup can be seen as a combined DA method of rotation and color constancy.

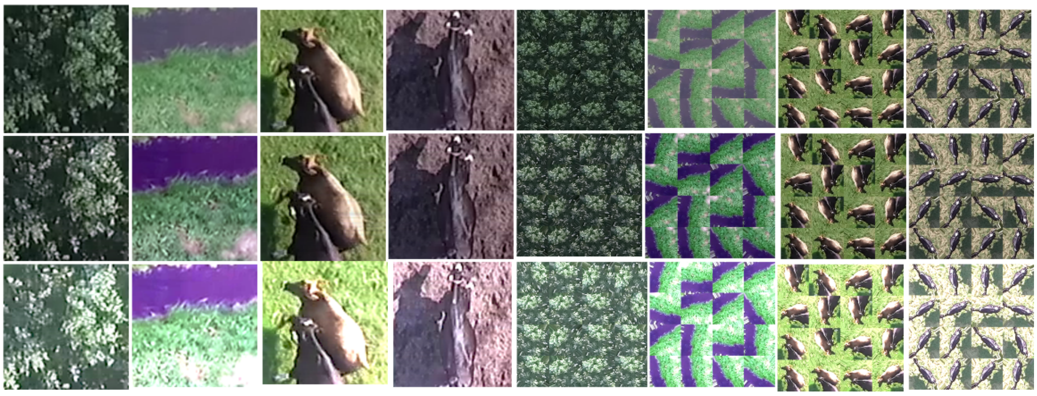


Figure 12: Examples of the original and rotation-matrix DA (ROT-DA) images from the Aerial UAV dataset, the first row accounts for the original images (column 1 - 4) and ROT-DA images (column 5 - 8) without color constancy. The second and the third rows are the MSR and MSRCR versions for both the ORIG and ROT-DA images respectively. Our proposed rotation matrix algorithm eliminates zero pixel values generated due to rotation by filling it with nearest neighbor pixels. The color constancy algorithm shows enhancement in the illumination and light intensities for each of the image samples.

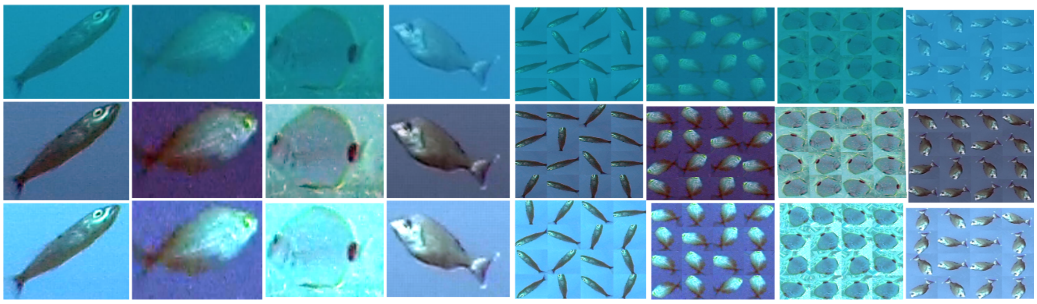


Figure 13: Examples of the original and rotation-matrix DA (ROT-DA) images from the Croatian fish dataset (Jaeger et al., 2015), the first row accounts for the original images (column 1 - 4) and ROT-DA images (column 5 - 8) without color constancy. The second and the third rows are the MSR and MSRCR versions for both the ORIG and ROT-DA images respectively. The color constancy algorithms also show improvement in the image resolution compared to the original image samples.

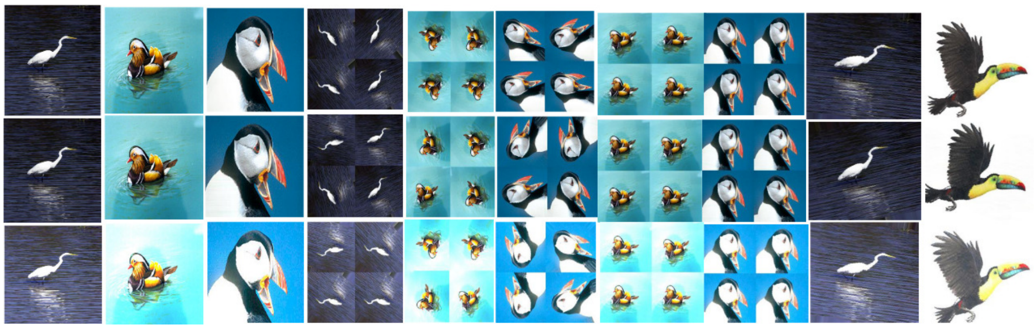


Figure 14: Examples of the original and rotation-matrix DA (ROT-DA) images, the first row accounts for the original images (column 1 - 3), 2 × 2-ROT-DA images using V1 rotation condition (column 4 - 6), 2 × 2-ROT-DA images using V2 rotation condition (column 7 - 8) and 1 × 1-ROT-DA images using V2 rotation condition (column 9 - 10) all mentioned without color constancy. The second and the third rows are the MSR and MSRCR versions for both the ORIG and ROT-DA images respectively.

4.2 Image Recognition Methods

4.2.1 CNN Architecture

The architecture used in this section is similar to that described in Section 3.2.1. The used network is directly derived from the famous GoogleNet architecture as proposed in (Szegedy et al., 2015). We eliminated all the layers after the inception 4a module, except for layers which lead to the first classifier and this is because the used datasets contain few classes (2, 6, and 12) for the Aerial UAV, Bird-600 and Croatia fish datasets respectively. Hence, we want to know how the reduced architecture can handle these problems.

4.2.2 CNN Experimental Setup

In this subsection, we explain the experimental setups used for each of the datasets.

4.2.2.1 *CNN Experimental Setup for the Aerial UAV Dataset*

In this dataset, the effective sizes of the train-validation sets of the variants of color-constancy-DA images in either original or rotation-matrix form are increased to double or three times the original dataset size for the different subsets of this dataset. The new versions of the datasets result in a slight modification of the CNN training parameters: changes in the solver test iterations (validation / train) for the respective datasets are detailed in Table 10. The table also shows the dataset distribution. Moreover, we employed similar experimental settings as explained before in 3.2.1.1. We remark that the test iterations for the three test sets that exist in either ORIG or ROT-DA alone were kept constant with the aim to examine the effectiveness of the new CNN models. Please note that we separated the rotation-matrix and original versions of the test sets before applying color constancy only on the train-validation sets.

Table 10: Dataset split information. For the Aerial UAV dataset the first four DA methods construct a dataset two times larger than the original dataset for all subfolds. STI means solver test iterations.

Dataset	Dataset Variants	Sub	Train	Val	Test	STI
UAV	ROT+MSR-ROT-DA	Sub 1	5950	1488	262	297
	ROT+MSRCR-ROT-DA	Sub 2	2259	565	2569	113
	ORIG+MSR-ORIG-DA	Sub 3	4529	1133	1150	226
	ORIG+MSRCR-ORIG-DA					
	ROT+MSRCR-ROT+MSR-ROT-DA	Sub 1	8925	2232	262	446
	ORIG+MSRCR-ORIG+MSR-ORIG-DA	Sub 2	3388	848	2569	169
		Sub 3	6793	1700	1150	340
Bird	ORIG, ROT-DA	5 folds	270	30	300	30
	ROT+MSRCR-ROT+MSR-ROT-DA	5 folds	810	90	300	90
	ORIG+MSRCR-ORIG+MSR-ORIG-DA	5 folds	810	90	300	90
Fish	ORIG, ROT-DA	5 folds	240	160	394	20
	ROT+MSRCR-ROT+MSR-ROT-DA	5 folds	720	480	394	60
	ORIG+MSRCR-ORIG+MSR-ORIG-DA	5 folds	720	480	394	60

4.2.2.2 *CNN Experimental Setup for Croatia Fish Dataset*

In this dataset, we investigated the ORIG and ROT-DA dataset alone, and color-constancy-DA of ORIG and ROT-DA separately. Moreover, we also studied the impact of grid resolution on the ROT-DA; this means we used 4×4 and 2×2 ROT-DA images in our experiments separately. Similar CNN experimental settings as described in subsection 4.2.2.1 were used. The additional modifications to the proposed CNN include; the batch size for training, validation, and testing is set to 12/8/1 respectively. The training of each of the CNN models uses maximum iterations of 7200, which generates a snapshot at each interval of 720 iterations, the step-size is set to 3600. This results in a decrease in the learning rate to $\frac{1}{10}^{th}$ times the base learning rate of 0.001. For the ORIG and ROT-DA alone we set the test interval to 240 while for the color-constancy DA versions (ORIG/ROT-DA) it is set to 720. The dataset variants were shuffled based on five-fold cross-validation with five different test sets ensuring no overlap exists in the train-validation sets. Please note that we separated the rotation-matrix and original versions of the test sets before applying color constancy only on the train-validation sets. The dataset distributions are detailed in Table 10.

4.2.2.3 *CNN Experimental Setup for Bird-600 Dataset*

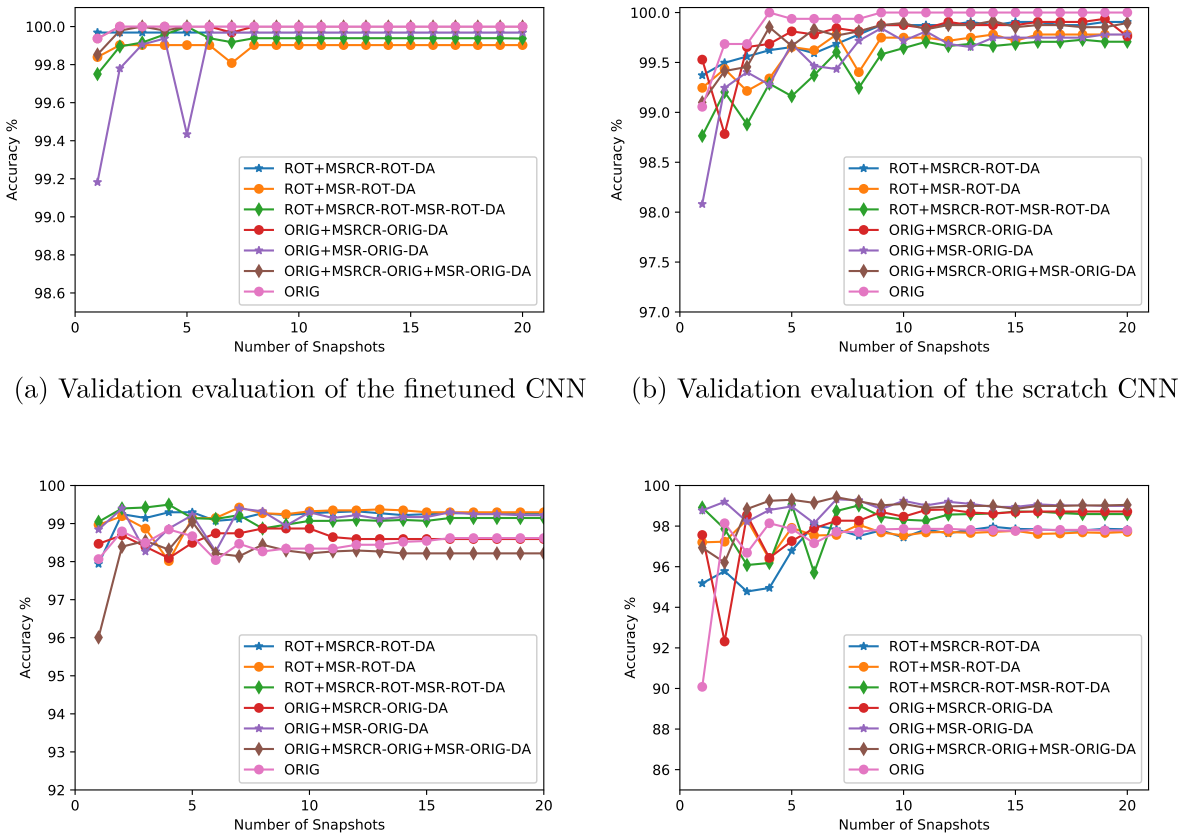
In this dataset, we investigated the ORIG and ROT-DA alone, and color-constancy-DA of ORIG and ROT-DA separately. Our preliminary experiments suggest that the 2×2 ROT-DA yields better performances as compared to the larger 4×4 grid. This informed our choice of this grid, so we will use smaller grids for this dataset. A similar CNN experimental setup as described in subsection 4.2.2.1 is used. The additional modification to the proposed CNN include; the batch size for training, validation, and testing is set to 9/1/1 respectively. The training of each of the CNN models uses maximum iterations of 8100, which creates a snapshot at each interval of 810 iterations, the step-size is set to 4000. We used a base learning rate of 0.001. For the ORIG and ROT-DA alone we set the test interval to 270 while for the color-constancy-DA versions (ORIG/ROT-DA) it is set to 810. Similarly, the various dataset variants were shuffled based on five-fold cross-validation with five different test sets ensuring no overlap exists in the train-validation set. Please note that we separated the rotation-matrix and original versions of the test sets before applying color constancy only on the train-validation sets. The dataset distributions are detailed in Table 10.

4.3 Results

This section entails the discussion of the classification performances on the used datasets.

4.3.1 Results on the Aerial UAV Dataset

The CNN training computing time on the color-constancy DA variants for the different subsets is $t \leq 46$ min. We used the same approach of computing the weighted mean of the accuracies for the 3 subsets as reported before. The sub-figures in Figure 15 shows the learning curves for both training and testing on the color-constancy-DA variants of ORIG and ROT images respectively. From Figure 15a and Figure 15b, we observe that CNN validation accuracies of the color-constancy-DA methods yield very similar performances for both finetuned and scratch experiments.



(a) Validation evaluation of the finetuned CNN

(b) Validation evaluation of the scratch CNN

(c) Test evaluation of the finetuned CNN

(d) Test evaluation of the scratch CNN

Figure 15: Weighted mean classification accuracy on the Aerial UAV Dataset (different color-constancy DA approaches) while training for 10K iterations (20 snapshots) using CNN with cross entropy loss function. Please note that not all graphs are visible due to overlap.

From Figure 15c, we observe that the use of finetuned CNN on the ROT-MSRCR-ROT+MSR-ROT-DA attained a peak accuracy of $\sim 99.5\%$ at the 4^{th} snapshot while that of finetuned CNN on the ORIG+MSRCR-ORIG+MSR-ORIG obtained $\sim$ 99.06% at the 5^{th} snapshot. In both approaches, the performances reduce for longer iterations; this suggests that early stopping will be most appropriate for these methods. The validation performance in Figure 15a, shows that most of the techniques examined were stable after the 7^{th} snapshot (3.5K iterations). Hence we choose this iteration point as the basis of our comparison. A summary of the validation and the test accuracies is reported in Table 11. Overall, the finetuned CNN applied on ROT+MSR-ROT-DA yields a very good performance for almost all iterative points of evaluation.

In this dataset, using finetuned CNN on color-constancy-DA with ROT images yields a higher accuracy than with the finetuned CNN using either color-constancy-DA with ORIG images or ORIG images alone. However, all finetuned CNN results obtained using color-constancy-DA images does not surpass results obtained from finetuned CNN on ROT-DA images alone. This is possibly due to fact that the test sets are only using ROT-DA images. Overall the proposed rotation-matrix algorithm leads to higher accuracies on this dataset with or without the color constancy algorithm.

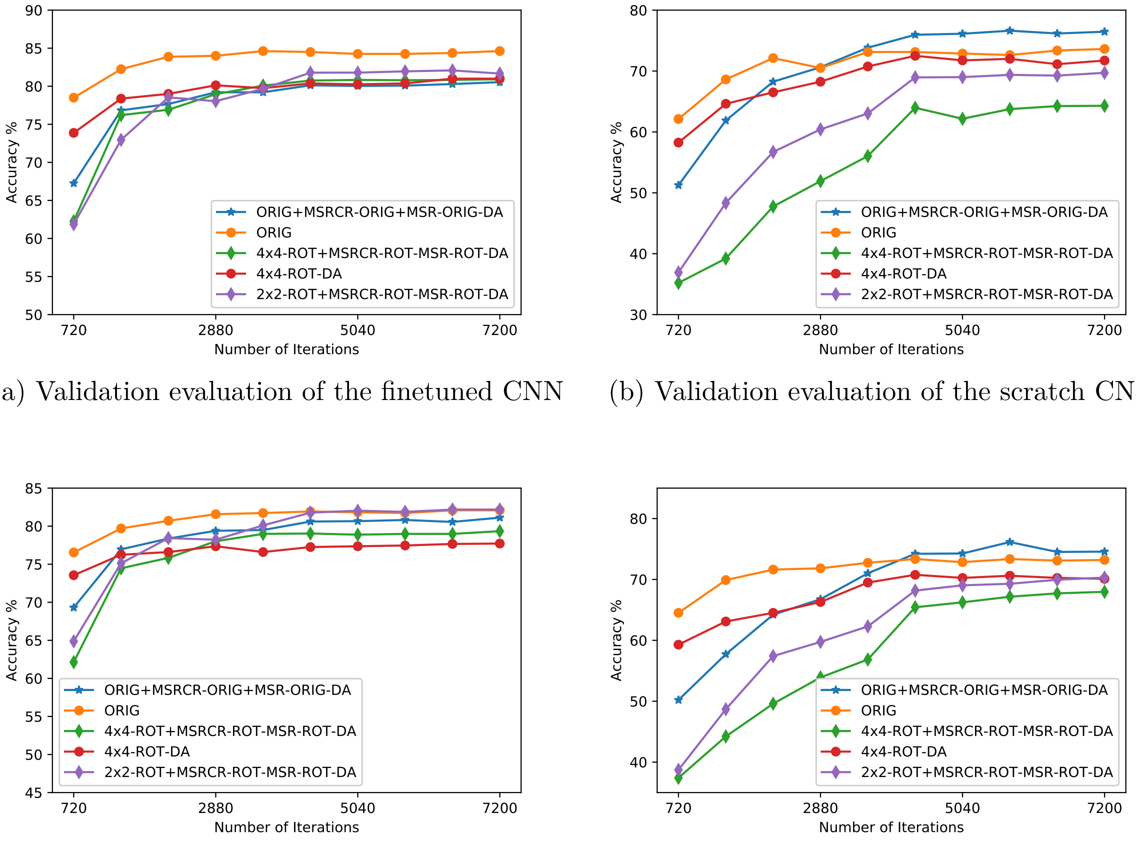
In contrast to this observation, in the scratch experiments, the results obtained from training scratch CNNs on color-constancy-DA with ORIG images outperforms CNN results obtained on ROT-DA and ORIG images alone. Thus it seems that adding more images to train the scratch CNNs plays the most important role. Based on this observation, we will use the best scratch technique (ORIG-MSRCR-ORIG+MSR-ORIG-DA) and its rotation-matrix version on the next two datasets. It is surprising that the scratch CNN performs better than the finetuned CNN on the ORIG+MSRCR-ORIG+MSR-ORIG-DA dataset. This may be caused by some overfitting problem, which we observed in the test accuracy of subset 2.

Table 11: Weighted mean of the validation and test classification accuracies of the CNN applied on different versions of the Aerial UAV dataset

Training CNN	Dataset-Variants	Validation	Test
Finetuned CNN	ROT-DA (Okafor et al., 2017)	99.94	**99.65**
	ROT+MSR-ROT-DA	99.81	99.42
	ORIG+MSR-ORIG-DA	99.97	99.40
	ROT+MSRCR-ROT+MSR-ROT-DA	99.91	99.22
	ROT+MSRCR-ROT-DA	99.97	99.12
	ORIG+MSRCR-ORIG-DA	99.97	98.74
	ORIG (Okafor et al., 2017)	100.00	98.67
	ORIG+MSRCR+ORIG+MSR-ORIG-DA	100.00	98.14
Scratch CNN	ORIG+MSRCR+ORIG+MSR-ORIG-DA	99.73	**99.41**
	ORIG+MSR-ORIG-DA	99.43	99.32
	ROT+MSRCR-ROT+MSR-ROT-DA	99.60	98.74
	ORIG+MSRCR-ORIG-DA	99.84	98.27
	ROT-DA (Okafor et al., 2017)	99.68	98.18
	ORIG (Okafor et al., 2017)	100.00	97.87
	ROT+MSRCR-ROT-DA	99.69	97.84
	ROT+MSR-ROT-DA	99.77	97.56

4.3.2 Results on Croatia Fish Dataset

We trained the CNNs using five-fold cross-validation data splits. The training time of the CNN models for each of the methods is $t \leq 16$ min. The models generated from the CNNs using color-constancy-DA variants with (ROT or ORIG) or (ROT or ORIG alone) were used to compute the accuracy on the test sets that contain either ORIG or ROT-DA images without color constancy. The learning curves for train-validation and testing phases while training for 7200 iterations are shown in Figure 16.



(a) Validation evaluation of the finetuned CNN

(b) Validation evaluation of the scratch CNN

(c) Test evaluation of the finetuned CNN

(d) Test evaluation of the scratch CNN

Figure 16: Five-folds cross-validation mean classification accuracy on the Croatia Fish Dataset while training for 7200 iterations using CNNs with the cross entropy loss function.

The mean accuracies for test and validation sets for the different approaches after that number of iterations are reported in Table 12. We report that there is no significant difference between the test and validation performances for most methods. This indicates that the test and validation performances are consistent.

From Table 12, we observe that the finetuned CNN on ORIG alone, the color-constancy-DA on ORIG and the 2 × 2-ROT version of the dataset all yield high accuracies. There is no significant difference in accuracies between these three methods. The best method is the finetuned CNN on the 2 × 2-ROT+MSRCR-ROT+MSR-ROT-DA variant of this dataset.

When we compare the results of the finetuned CNN applied on 2 × 2-ROT+MSRCR-ROT+MSR-ROT-DA to 4 × 4-ROT-DA, there exists a significant difference ($p < 0.05$). This indicates that the use of color constancy DA with ROT images and the right choice of grid resolution are important for this dataset. We also note that the finetuned CNN significantly outperforms the scratch CNN on this dataset.

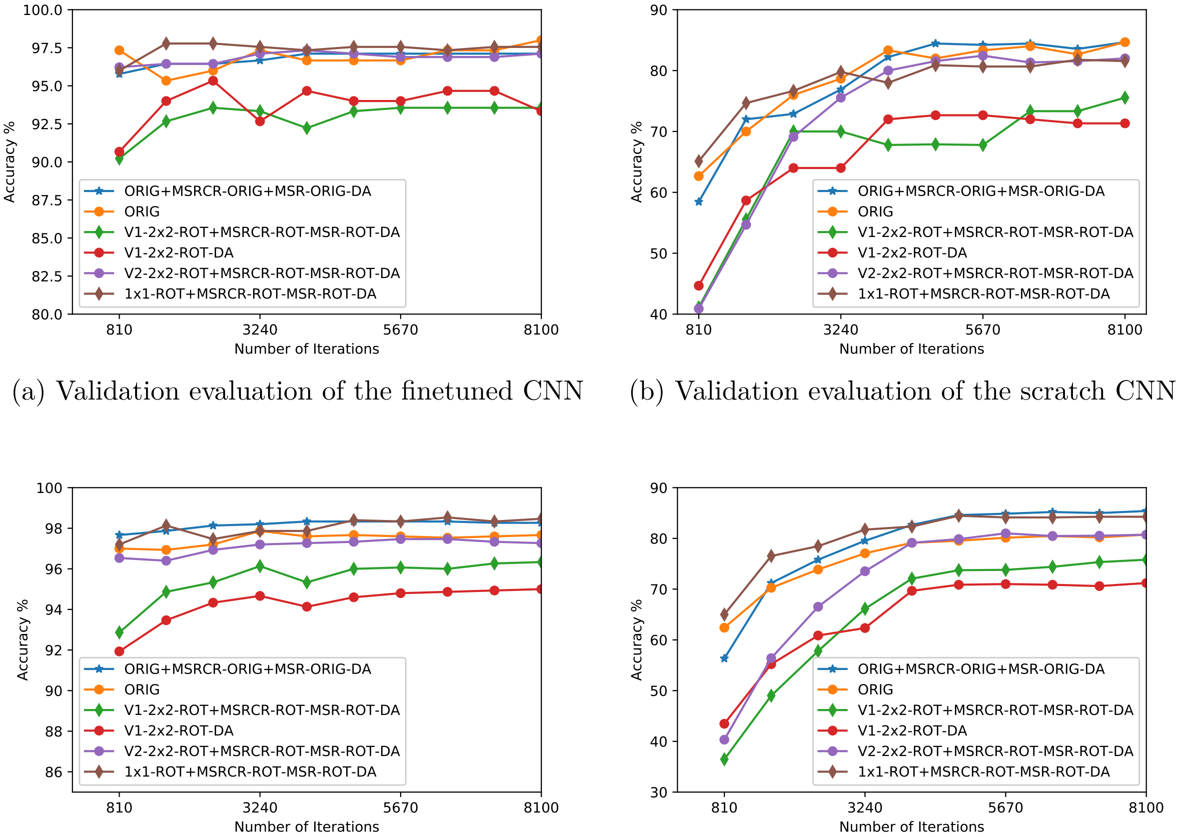
For the scratch experiments, training the CNN using ORIG+MSRCR-ORIG+MSR-ORIG-DA yields the highest accuracy. This best scratch CNN approach significantly outperforms the 4 × 4 ROT+MSRCR-ROT+MSR-ROT ($p < 0.05$). Overall, the choice of color constancy DA with 2 × 2 ROT images works better in our experiment than the use of color constancy DA with 4 × 4 ROT images.

Table 12: Five-fold cross-validation and test classification accuracies and standard deviations of the CNN applied on different versions of the Croatia fish dataset

Training CNN	Dataset-Variants	Validation	Test
Finetuned CNN	2 × 2-ROT+MSRCR-ROT+MSR-ROT-DA	81.67 ± 2.65	**82.18 ± 3.44**
	ORIG	84.63 ± 2.78	82.08 ± 4.21
	ORIG+MSRCR+ORIG+MSR-ORIG-DA	80.54 ± 2.81	81.12 ± 3.16
	4 × 4-ROT+MSRCR-ROT+MSR-ROT-DA	80.92 ± 2.82	79.34 ± 1.66
	4 × 4-ROT-DA	81.00 ± 1.66	77.72 ± 1.72
Scratch CNN	ORIG+MSRCR+ORIG+MSR-ORIG-DA	76.46 ± 2.40	**74.56 ± 4.10**
	ORIG	73.64 ± 1.99	73.19 ± 3.12
	2 × 2-ROT+MSRCR-ROT+MSR-ROT-DA	69.71 ± 3.43	70.30 ± 4.20
	4 × 4-ROT-DA	71.73 ± 3.08	70.10 ± 3.74
	4 × 4-ROT+MSRCR-ROT+MSR-ROT-DA	64.29 ± 1.30	67.97 ± 4.88

4.3.3 Results on Bird Dataset

We trained the CNNs using five-fold cross-validation data splits. The training time of the CNN models for each of the methods is $t \leq 13$ min. The models generated from the CNNs using color-constancy-DA variants with (ROT or ORIG) or (ROT or ORIG alone) of this dataset were used to compute the accuracies on the test sets that only contain either ORIG or ROT-DA images (without color constancy images). The learning curves for train-validation and testing phases, while training for 8100 iterations are shown in Figure 17.



(a) Validation evaluation of the finetuned CNN

(b) Validation evaluation of the scratch CNN

(c) Test evaluation of the finetuned CNN

(d) Test evaluation of the scratch CNN

Figure 17: Five-folds cross-validation mean classification accuracy on the Bird-600 Dataset while training for 8100 iterations using CNNs with the cross entropy loss function.

The mean accuracies for test and validation sets after that number of iterations are reported in Table 13.

From this table, we report that there is no significant difference between the test and validation performances for each of the examined methods, this shows again that the test and validation performances are consistent to each other. From the sub-figures in Figure 17, we observe that the

Table 13: Five-fold cross-validation and test classification accuracies and standard deviations of the CNN applied on different versions of the Bird-600 dataset

Training CNN	Dataset-Variants	Validation	Test
Finetuned CNN	1 × 1-ROT+MSRCR-ROT+MSR-ROT-DA	97.56 ± 2.47	**98.47 ± 0.34**
	ORIG+MSRCR+ORIG+MSR-ORIG-DA	97.11 ± 2.59	98.26 ± 0.25
	ORIG	98.00 ± 2.67	97.67 ± 0.56
	$V2 - 2 \times 2$-ROT+MSRCR-ROT+MSR-ROT-DA	97.11 ± 2.29	97.27 ± 0.93
	$V1 - 2 \times 2$-ROT+MSRCR-ROT+MSR-ROT-DA	93.55 ± 2.76	96.33 ± 0.79
	$V1 - 2 \times 2$-ROT-DA	93.33 ± 2.98	95.00 ± 0.73
Scratch CNN	ORIG+MSRCR+ORIG+MSR-ORIG-DA	84.67 ± 5.23	**85.40 ± 1.73**
	1 × 1-ROT+MSRCR-ROT+MSR-ROT-DA	81.56 ± 4.01	84.27 ± 1.58
	ORIG	84.67 ± 5.81	80.73 ± 2.73
	$V2 - 2 \times 2$-ROT+MSRCR-ROT+MSR-ROT-DA	82.00 ± 5.94	80.73 ± 1.51
	$V1 - 2 \times 2$-ROT+MSRCR-ROT+MSR-ROT-DA	77.11 ± 5.05	75.80 ± 1.15
	$V1 - 2 \times 2$-ROT-DA	71.33 ± 7.48	71.20 ± 3.03

finetuned CNNs outperform the scratch CNN methods on the different dataset variants.

The best techniques are the finetuned CNN on either 1 × 1-ROT+MSRCR-ROT+MSR-ROT-DA or ORIG+MSRCR+ORIG+MSR-ORIG-DA. These results indicate the importance of color constancy on the rotation-matrix or original images. This success can be attributed to training CNN weights with enhanced color information and with more images. To obtain this better performance, it was important to choose smaller rotational bounds $[-15^o, 15^o]$ as used in the 1 × 1-ROT+MSRCR-ROT+MSR-ROT-DA rather than the original rotational bounds $[1^o, 180^o]$. Such higher angular bounds may not be suitable for images that have an upright representation of objects.

Furthermore, we compared the results obtained with the finetuned CNNs on the different variants of this dataset to the baseline result from (Lazebnik et al., 2005) which obtained 92.33% using a probabilistic part-based method (maximum entropy framework). Our best approach significantly outperformed the baseline with a margin of 6.14% using the finetuned CNN on 1 × 1-ROT+MSRCR-ROT+MSR-ROT-DA. However, we remark that the obtained scratch CNN results on this dataset performed worse than the baseline method.

4.4 Discussion

In deep learning, data augmentation can play an important role if a dataset does not contain many training images. In this chapter, we used our novel data-augmentation method that transforms an image into a new image containing multiple random transformations of the image. We combined this method with the use of color constancy algorithms that add several transformed images to the training datasets. We created different combinations of methods: using original or rotation-matrix images combined with color constancy transformed images or not. These combinations were compared on three different animal datasets: Aerial UAV containing cows or not, a dataset with bird images, and a dataset with fish images. Overall we considered two broad forms of DA based on their increase (color-constancy-DA with ORIG or ROT-DA) or no increase (ROT-DA alone) in the amount of training images.

The results show that for the Aerial UAV dataset, the augmented rotation-matrix images are very useful. The Aerial UAV dataset consists of pictures taken from the sky, and therefore it is important to cope with 2D rotations to obtain the highest accuracies. It should be noted that this DA algorithm is useful for the CNNs, because although CNNs are more or less translational invariant, they are not rotational invariant. For the fish and birds dataset, the proposed rotation-matrix data augmentation method does not lead to better results than using the original images. For these datasets, the images show objects which are often in an upright position, and therefore there is less need to battle rotational variances.

The color constancy data augmentation helps in overall to get better accuracies, but the differences are not very large compared to using the original images. Only on the bird dataset, the color constancy data augmentation plays a very important role when training the CNN from scratch. The variation in colors is quite large for this dataset, and therefore adding additional images with different illumination levels is helpful. On this dataset, color constancy DA also improves the results of the finetuned CNN.

The results have also shown that the finetuned CNNs significantly outperform the CNNs trained from scratch on the Croatia fish and the

Bird-600 datasets. Furthermore, the finetuned CNNs obtain very high accuracies on the Aerial UAV and the Bird-600 datasets.

Future works can explore the use of deep neural network architectures to artificially transform colors in images. This could be done with a novel way of data augmentation or by adding initial layers that immediately transform the color pixels. It will also be interesting to create a deep neural network that can create the best rotation-matrix images, possibly trained using an adversarial learning framework.

5 ANALYSIS OF COLOR SPACES FOR IMAGE RECOGNITION IN DEEP LEARNING

This chapter explores the analysis of color space conversion on an image before applying deep neural networks for recognizing the images. Deep neural networks have obtained many successes for different image recognition problems. In most cases, the image datasets consist of images represented with red, green and blue (RGB) color channels. However, there are also image datasets consisting of shapes of objects that are represented in a binary black and white pixel format. This chapter attempts to examine if converting such datasets to other color spaces affects the performance of deep neural networks. For this aim, we developed a color conversion algorithm, which provides a framework for transforming both natural and binary-shape images to other color variants of the original images. For the experiments, we use two challenging shape datasets (MPEG-7 and Animal Shape) and a natural image dataset (Wild-Anim). To classify the images, we employed a reduced version of the GoogleNet convolutional neural network. On the Animal-Shape and MPEG datasets, the performance increase is very small as the CNN obtains high recognition accuracies using the original space. Finally, the results on the Wild-Anim dataset show that it is important to use the RGB color space for natural images to obtain the highest accuracies.

The central goal in computer vision is to construct algorithms that have the ability to recognize the semantics from an image. These algorithms face several challenges when dealing with shape and color variations in an image. One of the trending machine learning techniques which has the prowess to deal with these challenges is the use of convolutional neural networks (CNNs). There exist CNN architectures that have obtained very good performances on image recognition problems such as human face recognition (Pinto et al., 2011; Parkhi et al., 2015; Liu et al., 2015), handwritten character recognition (Ciresan et al., 2011), medical image recognition (Shin et al., 2016), pedestrian detection (Jiang et al., 2016), improvement of computational speed for object and character recognition (Gong et al., 2017), crowd control (Fu et al., 2015), and image retrieval (Montazer and Giveki, 2015). Some well-known CNN architectures are: AlexNet (Krizhevsky et al., 2012), GoogleNet (Szegedy et al., 2015), VGG-Net (Simonyan and Zisserman, 2014) and Residual Networks (ResNets) (He et al., 2016b). In this chapter, we will use the GoogleNet architecture and analyze the impact of artificially colorizing binary masked and natural image datasets.

An area of research which shares similarity compared to our proposed method is the development of colorization techniques. These techniques are mainly used to convert gray to color images. Early approaches have adopted several methods to approach the problem of colorization. In (Welsh et al., 2002; Gupta et al., 2012) the authors used automatic transfer techniques, while (Ironi et al., 2005) investigated the use of spatial voting with a global optimization method. The research by Levin et al. (2004) considered the use of a quadratic optimization algorithm in their study of colorization. Recently, some works have focused on colorization and enhancement of gray images using deep convolutional neural networks (CNNs) (Cheng et al., 2015; Zhang et al., 2016). However, no previous work has considered the conversion of binary masked (BW) or natural (RGB) images to other color variants of the original images for a given dataset, and then applying the CNNs to evaluate the classification performance on the different color variants of the used datasets.

Related to the datasets used in this chapter, several types of research have studied the use of segmentation-based algorithms or local feature descriptors for the recognition of segmented shape datasets. The joint learning framework approach (Ramesh et al., 2015) which combines several feature descriptors has been the state-of-the-art algorithm on the Animal-Shape dataset. This technique outperforms different segmentation-based methods (Bai et al., 2009; Li et al., 2011, 2010) and classical descriptors such as the combination of the bag of visual words with the histogram of oriented gradients and SIFT (HOG-SIFT-BOW) (Lim and Galoogahi, 2010) and the bag of words (BOW) alone (Ramesh et al., 2015).

In addition to the studies on the Animal-Shape dataset, we also examine MPEG-7 (Latecki et al., 2000), which is another challenging shape dataset. This dataset contains more classes compared to the Animal-Shape dataset and also has a high intra-class similarity. An early study by Sun and Super (2005) employed the combination of a segment-set and Bayesian technique for classification of this dataset. The research by Bai et al. (2009) showed that combining contour and shape features for shape classification significantly outperforms the use of single features such as contour segments or skeleton paths. The most recent research on this dataset is the use of region-based descriptors and the kernel-based extreme learning machine approach (Lin et al., 2017).

The concept of the classical computer vision methods is gradually becoming old-fashioned with the emergence of deep learning using different convolutional neural network architectures (Krizhevsky et al., 2012; Szegedy et al., 2015; Simonyan and Zisserman, 2014; He et al., 2016b). Previous studies have shown that the GoogleNet and AlexNet architectures perform better than classical feature descriptors (BOW or HOG-BOW) on a small version of the Wild-Anim dataset (Okafor et al., 2016) which exists in the RGB color space.

With this chapter, we want to understand the impact of artificially colorizing the binary masked (BW) or natural (RGB) images into other color variants before classifying them using customized GoogleNet architectures.

Contributions: This chapter describes the use of different versions of the GoogleNet architecture (fine-tuned and scratch instances) for investigating the classification performances on different color versions of image datasets. We propose a color conversion algorithm, which presents the following merits: 1) It can transform binary masked (BW) images to images represented in different color spaces (RGB, YCbCr, HSV, Lab), which may aid to boost the CNN's classification performance, and 2) It is an efficient algorithm and easy to implement or use.

The results show that by training a CNN on artificially colorized images or the original BW images, the classifier yields almost similiar performance levels on the Animal-Shape dataset. However, to the best of our knowledge, our system using either the new color variants or BW outperform all previous works on the Animal-Shape dataset. The success can be attributed to the used neural network system. For the MPEG-7 dataset, the performance improvement of using different color spaces is very small. Still, the results of the CNN architecture on the MPEG-7 dataset show that our system obtains the second best reported result on this dataset. Lastly, the results of the CNN applied on the Wild-Anim dataset show that the use of the natural RGB color space is important, as it obtains significantly better results than using other color spaces.

Outline: Section 5.1 describes the color-space conversion algorithm, the datasets used, and the visualization of the intensity analysis of some input images. Section 5.2 explains the instances of the customized CNN architectures and the experimental setups for each of the datasets. The experimental results of the CNNs on several variants of the used datasets are reported in section 5.3. Finally, the conclusion and some possible areas of future work are discussed in section 5.4.

5.1 Color Spaces and Datasets

This section explains the color space conversion algorithm, the datasets and preprocessing steps used in our experiments, and shows a clear visualization of the image intensities for the different color versions of an image taken from the Animal-Shape dataset.

5.1.1 The Color Spaces

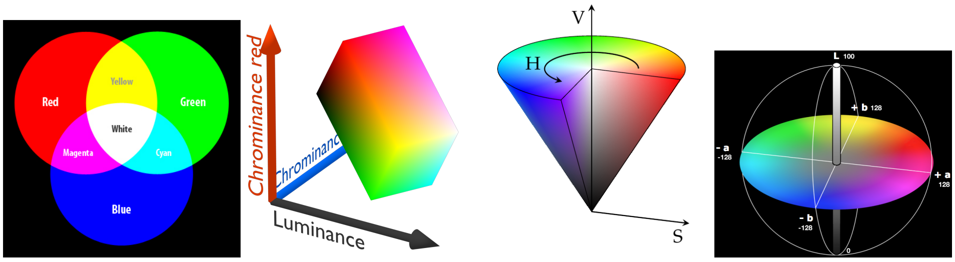
A color space is a method by which one can specify, create and visualize colors[1]. In the next subsections, the color components and the mathematical basis for each of the color spaces used in our study are described, followed by our newly developed color conversion algorithm.

[1] http://www.poynton.com/PDFs/coloureq.pdf

5.1.1.1 *Kinds of Color Spaces*

This subsection entails a brief discussion on the color spaces used in this paper. In total, we will use four different color spaces.

1. *RGB Color Space*: This color space is the most often used color space. It finds many application fields within the computer graphics and vision community. This color space consists of additive color components which are often represented based on the trichromatic theory[1]. The three main additive components consist of the colors red R, green G, and blue B. The combinations of the additive color components of an RGB color space are shown in Figure 18a and the different combinations (Jack, 2011) are presented in Table 14. Please note that the pixel values for the RGB color components lie between 0 and 255.



(a) RGB (b) YCbCr (c) HSV (d) Lab

Figure 18: Pictorial illustration of the different color spaces

Table 14: RGB color combinations

Color Component	Normal Range	White	Yellow	Cyan	Green	Magenta	Red	Blue	Black
R	0 to 255	255	255	0	0	255	255	0	0
G	0 to 255	255	255	255	255	0	0	0	0
B	0 to 255	255	0	255	0	255	0	255	0

2. *YCbCr Color Space*: There exist several equations which describe the YCbCr color space. We adopt the computer graphics YCbCr representation, which can be computed from the RGB color space

using the expression (Hsu et al., 2002; Bensaali and Amira, 2004) as defined in Equation 29:

$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 0.257 & 0.504 & 0.098 \\ -0.148 & -0.291 & 0.439 \\ 0.439 & -0.368 & -0.071 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} \tag{29}$$

The Y accounts for the luminance and is defined within a range $16 - 235$. The Cb and Cr account for the chrominance components with respect to blue and red color channels and these are defined within the range $16 - 240$ (Prathibha1 et al., 2012) as shown in Table 15. This color space can be represented as in Figure 18b. Furthermore, an 8-bit YCbCr pixel representation will be used that contains values between 0 and 255. This is required to avoid either overflow or underflow that may arise due to wrap-around (Jack, 2011).

Table 15: YCbCr color combinations adopted from (Prathibha1 et al., 2012) without normalization

Color Component	Normal Range	White	Yellow	Cyan	Green	Magenta	Red	Blue	Black
Y	16 to 235	235	210	170	145	107	82	41	16
Cb	16 to 240	128	16	166	54	202	90	240	128
Cr	16 to 240	128	146	16	34	221	240	110	128

3. *HSV Color Space*: This color space was created to manipulate colors. It finds application in the human perception and interpretation of color. In Figure 18c, we show a visual illustration of the HSV color space. Table 16 shows each component of the HSV color space. The Hue (H) component of this color space is tilted at a range from 0° to 360°, while the saturation (S) and value (V) components are normalized to a range between 0 and 1 (Jack, 2011).

Table 16: HSV color combinations

Color Component	Normal Range	White	Yellow	Cyan	Green	Magenta	Red	Blue	Black
H	0° to 360°	0°	60°	180°	120°	300°	0°	240°	0°
S	0 to 1	0	1	1	1	1	1	1	0
V	0 to 1	1	1	1	1	1	1	1	0

A modified mathematical equation used for computing the conversion from RGB to HSV (Chen et al., 2007; Liu et al., 2014) is defined as:

$$H = \begin{cases} 60\left(\frac{G-B}{max(R,G,B)-min(R,G,B)+\epsilon}\right) & if\ R = max(R,G,B) \\ 60\left(2+\frac{B-R}{max(R,G,B)-min(R,G,B)+\epsilon}\right) & if\ G = max(R,G,B) \\ 60\left(4+\frac{R-B}{max(R,G,B)-min(R,G,B)+\epsilon}\right) & if\ B = max(R,G,B) \end{cases} \tag{30}$$

$$S = 60\left(\frac{max(R,G,B) - min(R,G,B)}{max(R,G,B)+\epsilon}\right) \tag{31}$$

$$V = max(R,G,B) \tag{32}$$

Where $\epsilon > 0$ is a small constant. An 8-bit pixel image of the HSV color space can then be computed with:

$$H \to H/2, \qquad S \to 255 \times S, \qquad V \to 255 \times V \tag{33}$$

4. *Lab Color Space*: The Lab color space is a direct derivation from the $CIE\ XYZ$ color space. It can be defined as a non-linear representation of L, a, and b components which are intended to represent the logarithmic response of the eye[2]. This color space can be computed using the expressions below:

$$L = \begin{cases} 116(\frac{Y}{Y_n})^{\frac{1}{3}} - 16 & if \frac{Y}{Y_n} > 0.008856 \\ 903.3(\frac{Y}{Y_n}) & if \frac{Y}{Y_n} \leq 0.008856 \end{cases} \tag{34}$$

[2] http://www.poynton.com/PDFs/coloureq.pdf

$$a = 500\left(\frac{X}{X_n} - \frac{Y}{Y_n}\right) + \delta \tag{35}$$

$$b = 200\left(\frac{Y}{Y_n} - \frac{Z}{Z_n}\right) + \delta \tag{36}$$

$$\text{Where } \delta = \begin{cases} 128 & for \qquad 8 \;\; bit \\ 0 & for \qquad floating - point \end{cases}$$

X_n, Y_n, and Z_n denote the $CIE\ XYZ$ values for a given white reference point. X, Y, and Z represent the components of the XYZ color space which are derived from the RGB color space. In Figure 18d, we illustrate the color components of a Lab color space. The luminance component L is within the range from 0 to 100 between two opposite directions, and the chrominance (a and b) are within the range from -128 to 128 between two opposite directions for each of the chrominance components. Table 17 shows each component of the Lab space.

Table 17: Lab color combinations in floating-point pixels (without normalization)

Color Component	Normal Range	White	Yellow	Cyan	Green	Magenta	Red	Blue	Black
L	0 to 100	100.00	97.14	91.11	87.74	60.34	53.24	32.30	0.00
a	-128 to 128	0.00	-21.55	-48.09	-86.18	98.23	80.09	79.19	0.00
b	-128 to 128	0.00	94.48	-14.13	83.18	-60.83	67.20	-107.86	0.00

We remark that most of the images examined exist as 8-bit pixel intensities. An 8-bit representation of the Lab channels can be computed with:

$$L \rightarrow \frac{255L}{100}, \qquad a \rightarrow a + 128, \qquad b \rightarrow b + 128 \tag{37}$$

Where the Lab channels on the right-hand side of the equations correspond to; $0 \leq L \leq 100$, $-128 < a < 128$, $-128 < b < 128$

5.1.1.2 *Conversion Algorithm Between Color Spaces*

Based on the established mathematical relationship between the color spaces, it is possible to convert from RGB to other color spaces as was discussed in the previous subsection. The Caffe framework has a convert-imageset function, which is often used for creating the training, testing and validation sets, and stores each of these sets with an 'lmdb' data format. This function processes images in two ways; either as gray (1 channel) or color (3 channels). We did not use the gray option during our conversion, rather we processed all the images in each of the three datasets to a 3-channel representation irrespective of their original image information. This is because the fine-tuned GoogleNet architecture requires a 3-channel input image.

In this study, we convert the masked images that exist in BW (shape image datasets) and RGB (natural image dataset) to different color spaces: BW, RGB, HSV, Lab, and YCbCr. To understand how this was done, see the color-space conversion pseudocode in Algorithm 2. Some BW images have pixels which exist as 1 bit information. Hence direct color conversion will result in a plain background with the absence of a foreground (contour shape representation). We used a threshold function for mapping 8-bit into 1-bit pixel intensities, with the aim of preserving the contours within the masked images. Then, the resulting images are converted to indexed images using MatLab functions (`gray2ind`)[3] by specifying a color-map size that is set to 16. The indexed images are further converted using a MatLab function (`ind2rgb`) with a jet color-map size set to 16 to a true-color (RGB) with floating-point pixel values between $\{0, 1\}$. The true-color is multiplied by 255 to obtain pixel values in floating point (`double`) format. We further provide an intuitive mathematical explanation for the index to RGB image transformation.

First, the `ind2rgb` function computes a transformed index image defined as:

$$I_{tm} = max(1, min(I_m, size(C_m, 1))) \tag{38}$$

I_{tm} denotes the transformed indexed image given the index image I_m and color-map C_m as inputs. We used a jet color-map C_m that returns 16×3 color pixel values in floating point format. Note that both indexed

[3] http://nl.mathworks.com/help/images/ref/gray2ind.html

images have the same dimensions as the original input image with an image size of 250 × 250 pixels, but with different pixel information.

Secondly, each channel of the intended RGB color space is initialized to a zero array $0_{N\times M}$. The row N and column M of the initialized arrays have the same image size as the indexed image. The intended channels are mapped to each column of the described color-map C_m given I_{tm} as an input. The mapping transformation of I_{tm} to the specified color-map for each of the RGB color channels can be defined as:

$$R(:) = C_m(I_{tm}, 1), \qquad G(:) = C_m(I_{tm}, 2), \qquad B(:) = C_m(I_{tm}, 3) \tag{39}$$

Finally, the resulting channels were concatenated to form the RGB color space I_{rgb}:

$$I_{rgb} = [R; G; B] \tag{40}$$

The I_{rgb} image pixels all exist in a floating-point pixel format which lie in the range $[0, 1]$. We noticed that the image color information looks faded. Hence we scaled the pixels by multiplying the I_{rgb} image by a factor of 255. This improved the visibility of the image representation. Then the scaled image also existed as double. We employed the `imwrite` function to save the floating point (double) image into an 8-bit pixels image in .jpeg file format. This creates some color blending between the color channels so that object borders are smoothed a bit on a local scale. Still, on a global scale the overall impact of this is not large as can be seen in Figure 20.

Figure 19 shows an illustration of a patch pixel transformation pipeline from a BW image to an 8-bit RGB image. With the aim to obtain other color variants, the RGB images were transformed to the other color spaces using the mathematical transformation principles discussed in the previous subsection.

Additionally, we considered the conversion from RGB to BW images in the Wild-Anim dataset. This was achieved by first converting the RGB to gray-scale images and then processing the gray into BW pixels by applying a threshold to the image pixels. The BW image is expected to have image pixels which exist either in 1-bit $\{0, 1\}$ or 8-bit $\{0, 255\}$. This means that if the pixel values are $\{1, 255\}$ (white) they represent high luminance greater than a threshold and other pixels are converted to 0 (black). In

Algorithm 2 Color Space Conversion Algorithm

Input : BW or RGB raw images $I = I_B(x, y, c)$ from an input directory, c is the number of channels, x, y are the pixel row and column, respectively.
Output : Converted output of the new color spaces $y = I_C(x, y, c)$ that is stored into an output directory.

```
 1: procedure CREATE A FILELIST OF ALL $N$ IMAGES WITHIN THE INPUT DIRECTORY AND SPECIFY THE SELECTION OF COLOR SPACE OPTIONS: RGB, HSV, YCBCR, LAB, BW
 2:     for each image $i \in N$ do
 3:         Check the image channels
 4:         if $c = 1$ and color-selection = $others$ then the image is BW
 5:             Check the BW image pixels,
 6:             If image pixels is 1-bit, then map to an 8-bit pixel $g \in \{0, 255\}$.
 7:             Otherwise pixels exist as 8-bit.
 8:             Transform the BW image to an indexed image $I_m$ with a 4-bit pixel.
 9:             The indexed image is transform to a true color image $I_{rgb}$ using eqn 40.
10:             The $I_{rgb}$ image in the floating-point format is upscaled by a factor of 255 to obtain higher intensity information across the color channels.
11:             If selected color space is RGB save into the output directory
12:             Otherwise, check the chain of $elseif$ statement to convert from RGB to other color space { YCbCr, HSV, Lab }
13:             If the condition of either step 11 or 12 is met, proceed to step 21.
14:         else $c = 3$ and and color-selection $\neq BW$
15:             If an image is RGB proceed to step 21.
16:             Otherwise, check statement to convert from RGB to other color space.
17:             $Elseif$  $c = 3$ and color-selection = $BW$
18:             Convert the RGB image into a gray image.
19:             Convert gray image to BW image with a threshold of 0.4
20:         end if
21:         Store all output images of the new color space into the output directory with pixel values existing as 8 bit pixels.
22:     end for
23: end procedure
```

our study, the Wild-Anim dataset was originally RGB. We converted the color pixels (RGB) to BW images in this dataset by adopting a threshold setting of 0.4, because it seemed to retain most information of animal shapes in the Wild-Anim images. The computer program of Algorithm 2 was developed in MATLAB. The new versions of these datasets were fed as input to the customized versions of the GoogleNet architectures.

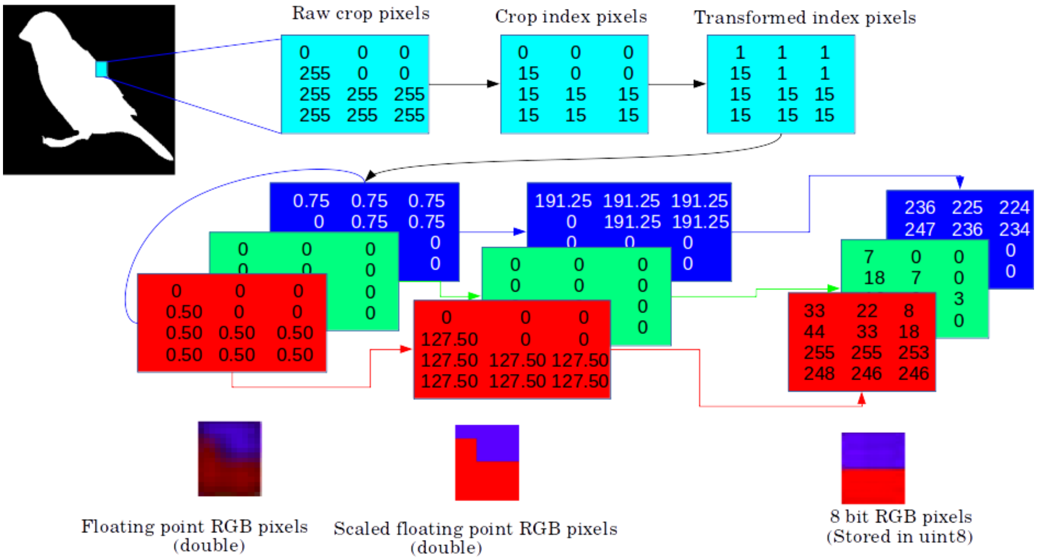


Figure 19: Transformation pipeline from BW pixels to 8-bit RGB pixels of an Animal-Shape image.

5.1.2 Datasets and Preprocessing

This section describes the various datasets used in our experiments, namely Animal-Shape, MPEG-7, and Wild-Anim.

5.1.2.1 *Animal-Shape Dataset*

This dataset has often been called either Animal or Shape dataset[4] by previous authors (Bai et al., 2009; Li et al., 2011, 2010; Ramesh et al., 2015). We refer to it as the Animal-Shape dataset. The Animal-Shape dataset was first presented in (Bai et al., 2009). The dataset has become a

[4] https://sites.google.com/site/xiangbai/animaldataset

famous benchmark dataset used for validating segmentation and shallow image-recognition techniques. This dataset contains masked images (BW) with 20 classes of different shapes of animals. In this dataset, there are in total 2000 images, and in each class there exist 100 individual images which are positioned in different orientations. The images are in 8-bit pixel format with non-uniform image sizes. We normalized the image sizes to 250 × 250 pixels. Algorithm 2 is used for converting the BW images into the different color spaces with the aim to create new versions of the dataset. Some example pictures for the original and the converted versions of this dataset are shown in Figure 20.

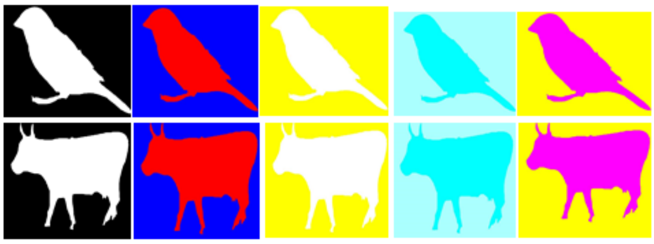


Figure 20: Example images of BW conversion to different color variants from the Animal Shape dataset. From left to right: BW (original), RGB, Lab, HSV and YCbCr.

5.1.2.2 *MPEG-7 Dataset*

MPEG-7 (Latecki et al., 2000) is another challenging shape dataset. This dataset contains a total of 1400 images and has 70 classes with 20 individual images per class. The images within this dataset exist as BW either in 1-bit or 8-bit pixel format with non-uniform image sizes. Again, we normalized the image sizes to 250 × 250 pixels. We report the data split for this dataset and the CNN experimental settings in subsection 5.2.2.2. We employed Algorithm 2 to obtain new color versions of this dataset. Some example pictures for the original and the converted versions of this dataset are shown in Figure 21.

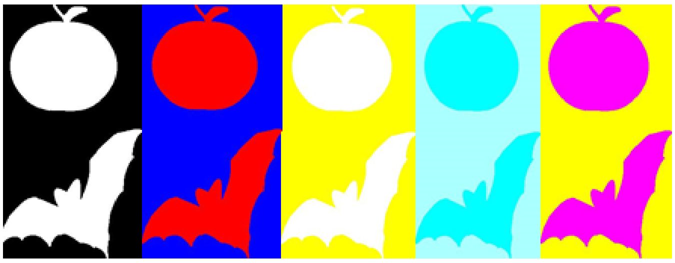


Figure 21: Example images of BW conversion to different color variants from the MPEG-7 dataset. From left to right: BW (original), RGB, Lab, HSV and YCbCr.

5.1.2.3 *Wild-Anim Dataset*

The Wild-Anim dataset is a new animal dataset originally presented in (Okafor et al., 2016). This dataset consists of a total of 5,000 images of 5 classes with 1000 images per class. The images present in this dataset are RGB images. We carried out experiments on a 20% subset[5] of the images within this dataset that contains 1000 images. We report the data split for this dataset and the CNN experimental settings in subsection 5.2.2.3. We used Algorithm 2 to obtain new color versions of this dataset. Some example pictures for the original and the converted versions of this dataset are shown in Figure 22.

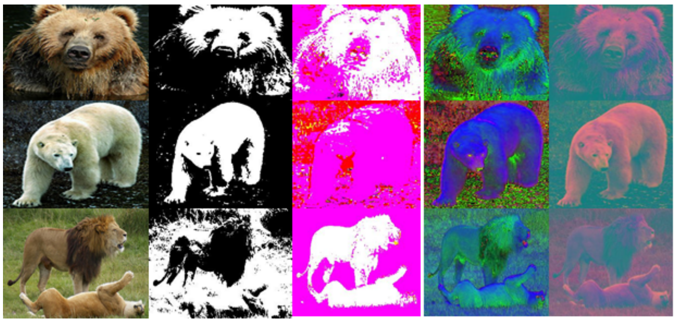


Figure 22: RGB Wild-Anim dataset conversion to other color spaces. From left to right: RGB (original), BW, Lab, HSV and YCbCr.

[5] http://www.ai.rug.nl/~emmanuel/wild-animdataset.html

5.1.3 Intensity Analysis on Color Variants of an Animal-Shape Image

In this subsection, we want to understand and visualize the image pixel intensities for the original and new versions of the Animal-Shape dataset. For this aim, we used the 'bird1.tif' as shown in Figure 23 to illustrate the pixel intensity for the channels in each of the color spaces. The original image size is 324×640 pixels with an aspect ratio (`AR`) of ~ 1.98. We rescaled the image size to 250×250 pixels. Hence resulting in an `AR` of 1 and this implies that $\sim 49.5\%$ of the `AR` from the original image is reduced. Furthermore, the rescaling introduced slight anamorphic distortions. With the aim to visualize each color channel, we computed the log value of the pixel counts as shown in Figure 24. The pixel values in the original BW image only exists as $\{0, 255\}$ and most of the saturation is at 0 as shown in Figure 24a. Because of the `imwrite` function the newly obtained images for the different color spaces consist of all 3 color components.

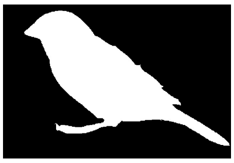
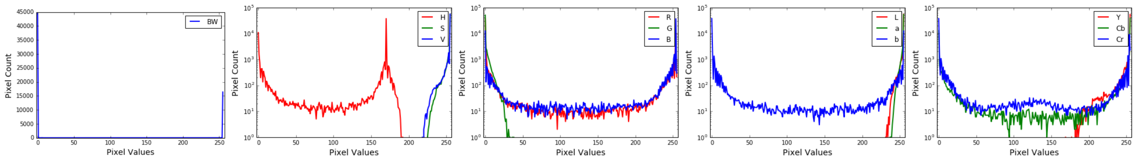


Figure 23: Example image used for intensity visualization of the Animal-Shape dataset



(a) (b) (c) (d) (e)

Figure 24: Pixel intensity analysis of the different color spaces of an Animal-Shape image (bird1.tif). The pixel count for each of the color variants is represented in log-scale: (a) Original BW Image (8-bit), (b) 8-bit HSV Image, (c) 8-bit RGB Image, (d) 8-bit Lab Image, and (e) 8-bit YCbCr Image

We observed that the channel $\{B, b, Cr\}$ in most of the color spaces $\{RGB, Lab, YCbCr\}$ shows a better intensity spread as shown in Figure 24(c) to 24(e), compared to other channels within the mentioned color spaces. An exception to this rule is the H channel of the HSV, whose maximal pixel intensity lies around 165 as shown in Figure 24(b). Note that the H channel has values between 0 and 180.

5.2 Deep Learning Setup

We study deep learning using convolutional neural networks (CNNs) to deal with the discussed datasets. We describe in this section several instances of the GoogleNet architecture and our experimental setup.

5.2.1 Instance of GoogleNet

A brief explanation of this technique was discussed in subsection 2.2.2.

5.2.1.1 *Scratch GoogleNet*

The Scratch GoogleNet architecture does not rely on any pre-trained CNN model but is trained using randomly initialized weights. Scratch GoogleNet employs Xavier Initialization (Glorot and Bengio, 2010). This algorithm allows initializing learnable weights from an initial distribution. We considered two instances of the Scratch GoogleNet architecture based on the numbers of filters in each convolutional layer within the last inception module.

Original Scratch GoogleNet (OS-GoogleNet) This version of the CNN contains a max-pooling layer and six convolutional layers. The original number of filters (neurons) in each convolutional layer within the last inception layer is as follows: 384, 192, 384, 48, 128 and 128 respectively. Our proposed system setup with a simplified representation of the used CNN (Szegedy et al., 2015) is shown in Figure 25.

Reduced Scratch GoogleNet (RS-GoogleNet) The RS-GoogleNet is derived from the OS-GoogleNet by reducing the numbers of filters in the convolutional layer within the last inception layer of the OS-GoogleNet. This method is motivated from the success obtained in the research paper in (Okafor et al., 2016). In this study, we want to exploit the efficacy of this method on the different color variants of the datasets used. This version of the CNN is designed to contain the following numbers of filters in each convolutional layer of the last inception module: 384, 24, 24, 24, 16 and 16 respectively. The block diagram illustrating this modification is shown in Figure 2.

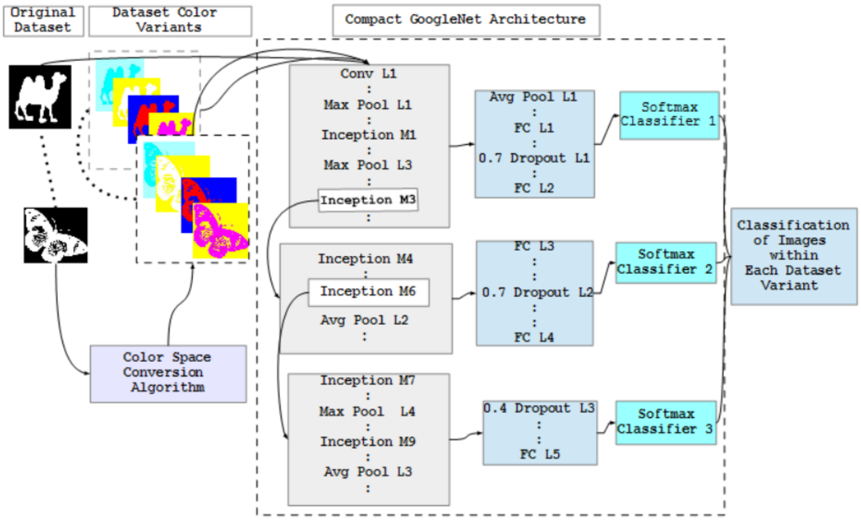


Figure 25: Block diagram showing our system setup using a simplified version of the GoogleNet architecture (Szegedy et al., 2015) applied on the MPEG-7 dataset; the ':' denote other modules or network layers.

5.2.1.2 *Finetuned GoogleNet*

In this subsection we will consider two instances of this architecture.

Original Finetuned GoogleNet (OFT-GoogleNet) The OFT-GoogleNet is a CNN that requires the use of pretrained weights from the ImageNet dataset. The pretrained model is used to initiate the training process using our CNN configurations.

Reduced Finetuned GoogleNet (RFT-GoogleNet) This is similar to to the OFT-GoogleNet with the only difference that it is trained using the

RS-GoogleNet configuration. So the training starts with pretrained weights rather than using randomly initialized weights for the reduced architecture.

5.2.2 Experimental Settings

Our experimental procedure is similar to that of (Szegedy et al., 2015) that employs stochastic gradient descent (Krizhevsky et al., 2012) and the momentum update rule on the weights as described in Equations 41 and 42.

$$u_{i+1} = \mu u_i - \alpha_L \left(\delta W_i + \left(\frac{\partial L}{\partial W_i} \right)_{D_i} \right) \tag{41}$$

$$W_{i+1} = W_i + u_{i+1} \tag{42}$$

where L is the cross-entropy loss function, μ is the momentum value, α_L is the learning rate, δ is the value for weight decay, u_i is the momentum variable, i is the iteration number, D_i is the batch over index iteration i and $\left(\frac{\partial L}{\partial W_i}\right)$ computes the mean over the i^{th} batch D_i of the derivative of the objective function with respect to W_i.

The various instances of the GoogleNet in our study use a simple data augmentation technique that involves cropping and flipping (horizontal reflection). This is done to create additional examples of images during online training of the CNN. For the experiments, we employed the following uniform parameters: we trained each scratch model (OS-GoogleNet/RS-GoogleNet) for $30,000$ iterations, while the fine-tuned models (OFT-GoogleNet/RFT-GoogleNet) were trained for $10,000$ iterations. All training of the CNN models was carried out on a Ge-Force GTX 960 NVIDIA GPU. The overall training time of the CNN models is in the range $0.3 \leq t \leq 5.6$ hours for each run, depending on the dataset used and the corresponding CNN setup. The CNNs employ a crop size of 224×224 pixels, momentum 0.9, interval display and average loss were set to 40, the power value is set to 0.5 with a gamma value that is set to 0.1 and the weight decay is set to 0.0002. The maximum iterations and step-size were set to $10,000$ and $30,000$ for finetuned and scratch experiments respectively. During training of the CNN models, we employed a fixed learning rate. The other CNN parameters and data distributions in our study are dependent on the dataset used.

5.2.2.1 *Animal-Shape Dataset Distribution and CNN Parameters*

We employed the same data-split as described in (Ramesh et al., 2015), whereby the train to test images are distributed in the ratio 50% : 50%. Furthermore, we split the training images into training/validation sets in the ratio 40% : 10%. This data distribution was shuffled 5 random times for each dataset. Additionally, we carried out two sets of experiments on this dataset.

Experiment I: We employ a base learning rate of 0.001, training and validation sets batch size was set to 10, the solver iterations parameter is set to 20, and each snapshot model of the CNN is created after every 1000 iterations. This implies that the finetuned and scratch CNN experiments require 10 and 30 snapshots respectively based on the already described number of maximum iterations. Please note that the train-validation sets are used for creating the CNN models. These models are then used for evaluating the different test sets. We used an evaluation test iteration value that is set to 100.

Experiment II: We also considered a different learning rate that is set to 0.002, using a higher number of the training batch size that is set to 20 and we reduced the validation batch size to 5. These settings resulted in a change in the solver iteration to a value 40. Moreover, we adjusted the number of snapshots to be generated after every 2000 iterations. Hence, the corresponding number of snapshots is 5 and 15 for finetuned and scratch CNN experiments respectively. We used an evaluation test iteration value that is set to 200.

5.2.2.2 *MPEG-7 Dataset Distribution and CNN Parameters*

We adopted the same data-split as described in (Bai et al., 2009; Lin et al., 2017), whereby the train and test images are distributed in the ratio 50% : 50% respectively. Furthermore, we split the training images into training/validation sets in the ratio 40% : 10% respectively. This data distribution was shuffled 5 random times for each dataset.

Experiment: The base learning rate used is 0.001, the solver iterations is set to 28, and the training/validation batch sizes of images is set to 20/5 respectively. Again each snapshot model is created at every 2,000 iterations. This results in 5 and 15 snapshots for finetuned and scratch experiments respectively based on the maximum iteration specification. We used an evaluation test iteration value that is set to 140.

5.2.2.3 *Wild-Anim Dataset Distribution and CNN Parameters*

We adopted the same experimental settings as described in (Okafor et al., 2016), the training/testing/validation image samples are distributed in the ratio 80% : 10% : 10% on 1000 examples of the dataset.

Experiment: We employ a base learning rate of 0.001, training and validation sets batch size was set to 10, the solver iterations is set to 10, and each snapshot model of the CNN is created after every 1000 iterations. This implies that the finetuned and scratch experiments require 10 and 30 snapshots respectively based on the already described number of maximum iterations. We used an evaluation test iteration value that is set to 10.

Please note that the number of output neurons from each of the last fully connected layers before the three classifiers in each of the CNN architectures is equal to the number of classes present in each of the datasets under study. Then the softmax classifier is used to make the prediction using the activations of the last output neurons.

5.3 Results

We report experimental results on both shape and natural image datasets. We trained using the train-validation sets to create several models that are used to compute the accuracies of the methods on the test sets using different color variants of the datasets. We report Top-1 classification accuracies for each of the methods examined on all the datasets used. Additionally, we report Top-5 classification accuracies on MPEG-7, this is because the dataset contains more classes.

5.3.1 Evaluation on the Animal-Shape Dataset

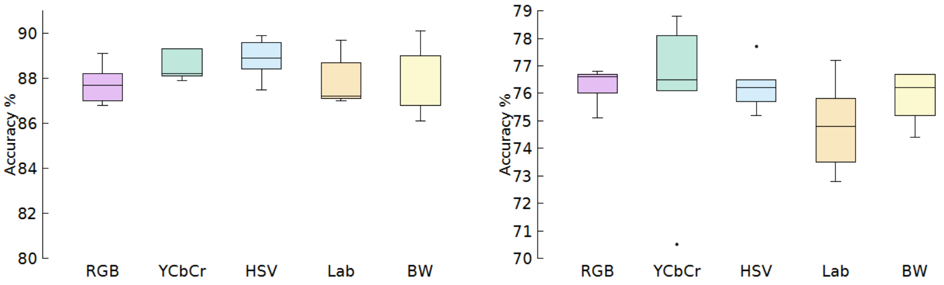
In our preliminary investigation, we employed all the instances of GoogleNet as described in section 5.2, to evaluate the CNNs on the masked images (BW) of the Animal-Shape dataset. We used the train-validation sets for training and testing of the CNNs based on five-fold cross-validation. We computed the mean of the test and validation classification accuracies with the corresponding confidence intervals for the two sets of experimental settings as described in the previous section. The results of our findings are presented in Table 18. From this table, we are 95% confident that there is no clear difference between the original and the reduced instances of GoogleNet. This observation motivated our use of the reduced instance of both scratch and finetuned versions of GoogleNet for the remaining experiments on other color variants of this dataset. Since the reduced instances of GoogleNet are direct alternatives to the original versions, we repeated the same CNN training procedure as described for the BW version of this dataset for all the color variants.

The test evaluation results for the reduced instances of scratch and finetuned CNN architectures are shown in Figure 26 and 27 for the two sets of experimental settings.

Table 18: Five-fold cross validation results showing accuracies (95% confidence Intervals) for the test and validation sets using different instances of GoogleNet on the BW version of the Animal-Shape dataset.

	Evaluation Type	OFT-GoogleNet	OS-GoogleNet	RFT-GoogleNet	RS-GoogleNet
Experiment I	**Test**	87.62 ± 1.40	76.74 ± 0.86	88.20 ± 1.32	75.84 ± 0.79
	Validation	88.40 ± 2.31	77.20 ± 1.06	89.40 ± 1.37	78.40 ± 2.31
Experiment II	**Test**	88.70 ± 0.94	76.80 ± 0.32	88.52 ± 0.54	77.02 ± 0.94
	Validation	90.30 ± 1.85	77.50 ± 2.06	90.70 ± 1.20	78.60 ± 2.83

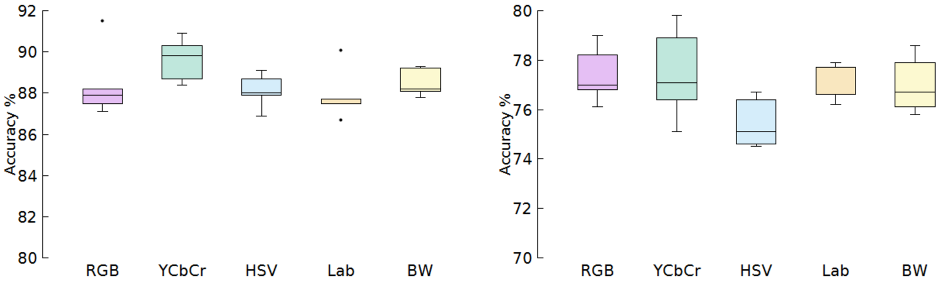
These figures show box-plot test distributions after $10K$ and $30K$ iterations for reduced finetuned and scratch experiments respectively on



(a) RFT GoogleNet Evaluation

(b) RS GoogleNet Evaluation

Figure 26: Reduced GoogleNet test accuracies for different color variants of the Animal-Shape dataset after 10K iterations for RFT-GoogleNet and 30K iterations for RS-GoogleNet (Experiment I).



(a) RFT GoogleNet Evaluation

(b) RS GoogleNet Evaluation

Figure 27: Reduced GoogleNet test accuracies for different color variants of the Animal-Shape dataset after 10K iterations for RFT-GoogleNet and 30K iterations for RS-GoogleNet (Experiment II).

the different color variants of this dataset. We computed the mean of the test and validation classification accuracies with the corresponding confidence intervals based on five-fold cross-validation. Our findings are reported in Table 19. The results show that the two experimental setups present almost similar levels of performance.

As the first remarkable observation from Table 19, the results show that the reduced instances for both kinds of GoogleNet architectures applied on some of the color variants show similar performance compared to the result obtained using GoogleNet on the BW version of this dataset. In addition, we observe that the fine-tuned versions of the CNN significantly

Table 19: Five-fold cross validation results showing the mean accuracies and 95% confidence intervals for the test and validation sets using reduced instances of GoogleNet applied on the Animal-Shape dataset.

Evaluation	CNN Method	RGB	YCbCr	HSV	Lab	BW
Experiment I: Test	RFT-GoogleNet	87.76 ± 0.73	88.56 ± 0.54	**88.86 ± 0.75**	87.94 ± 0.95	88.20 ± 1.32
	RS-GoogleNet	76.24 ± 0.56	76.00 ± 2.56	76.26 ± 0.74	74.82 ± 1.38	75.84 ± 0.79
Experiment II: Test	RFT-GoogleNet	88.44 ± 1.38	**89.62 ± 0.83**	88.12 ± 0.66	87.90 ± 1.01	88.52 ± 0.54
	RS-GoogleNet	77.42 ± 0.91	77.46 ± 1.49	75.46 ± 0.80	77.00 ± 0.59	77.02 ± 0.94
Experiment II: Validation	RFT-GoogleNet	87.10 ± 1.70	89.30 ± 1.34	89.20 ± 1.68	88.30 ± 2.64	90.70 ± 1.20
	RS-GoogleNet	78.40 ± 1.34	79.20 ± 2.05	77.60 ± 2.59	78.30 ± 3.32	78.60 ± 2.83

outperform all the scratch versions. Based on the results obtained, it is evident that the GoogleNet architecture applied on some of the color versions of this dataset profits from the pre-trained CNN model.

Furthermore, the best results in either experimental settings I or II for each method were compared to results obtained in previous research. The comparison of these methods is shown in Table 20. The results show that RFT-GoogleNet applied on the YCbCr version of this dataset outperforms the joint learning approach (Ramesh et al., 2015) with a significant improvement of 3.62%. In addition, the RFT-GoogleNet on HSV, BW, RGB, and Lab variant of this dataset also show a significant improvement compared to the best previous result with an improvement margin of 2.86%, 2.52%, 2.44%, and 1.94% respectively. The use of the scratch versions of the GoogleNet architecture applied on all the different color spaces showed a worse performance compared to most of the existing works.

5.3.2 Evaluation on the MPEG-7 Dataset

For this dataset we used the train-validation sets for training and testing the CNNs based on five-fold cross-validation. The models generated were used to evaluate the CNNs on each of the color variants of this dataset. The test distribution performances obtained using reduced instances of the GoogleNet architecture for Top-1 accuracies are shown in Figure 28. Furthermore, we compute the mean for the five-fold cross-validation and test accuracies with 95% confidence intervals for Top-1 and Top-5 classification accuracies. The results are reported in Table 21. From this

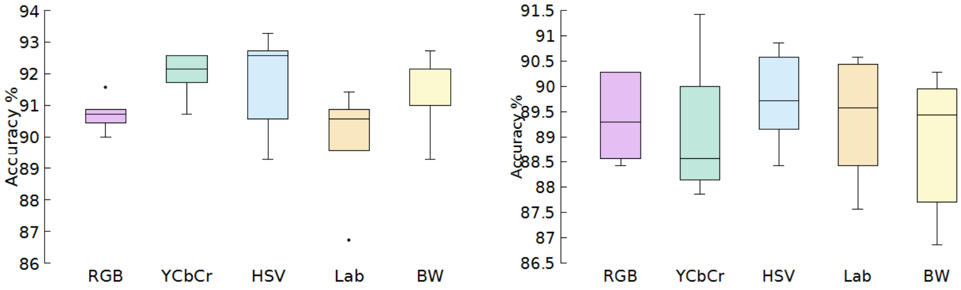
Table 20: Comparison of our approaches to other methods on the Animal-Shape dataset.

Methods	Test Accuracy
Joint Learning Approach (Ramesh et al., 2015)	86.0
Bi-gram (Ramesh et al., 2015)	83.5
BOW (Ramesh et al., 2015)	81.1
Contour Segment-Skeleton Path-Discriminative Path (CS-SP-DP) (Li et al., 2011)	80.7
HOG-SIFT-BOW (Lim and Galoogahi, 2010)	80.4
Shape-Tree (Li et al., 2010)	80.0
Contour Segment-Skeleton Path (CS-SP-IDSC-F) (Bai et al., 2009)	78.7
Contour Segment-Skeleton Path (CS-SP) (Bai et al., 2009)	78.4
RFT-GoogleNet-YCbCr	**89.62**
RFT-GoogleNet-HSV	**88.86**
RFT-GoogleNet-BW	**88.52**
RFT-GoogleNet-RGB	**88.44**
RFT-GoogleNet-Lab	**87.94**

table, we observe that there exist very close performance correlations between the various methods under study. Furthermore, the table also shows that the results obtained using RFT-GoogleNet on the YCbCr dataset are better than the CNN results on the original masked dataset (BW) with a margin of 0.71% and 0.51% for Top-1 and Top-5 accuracies respectively. Another good approach is the use of RFT-GoogleNet applied on the HSV version of the dataset that outperforms the original BW masked dataset with a difference of 0.45% and 0.60% for Top-1 and Top-5 accuracies respectively. The results obtained using the reduced scratch version of GoogleNet applied on all the color variants of this dataset outperforms the CNN trained on the original masked dataset (BW) for both Top-1 and Top-5 accuracies. However, the differences with respect to the Top-5 accuracies are very marginal. We also remark that the use of Lab and RGB color spaces did not work so well using the reduced fine-tuned GoogleNet when compared to the results obtained using the same instance applied on the original dataset (BW).

We compared our best methods to the previous works on this dataset as reported in Table 22. According to the results our Top-1 performance using RFT-GoogleNet on YCbCr and HSV are currently the second best results reported on this dataset. Our proposed Top-1 method even outperforms the most recent research (Lin et al., 2017) on this dataset as shown in

Table 22. The current state-of-the-art technique employs the use of contour segment and skeleton path (Bai et al., 2009) in their research. However, it is not very explicit what classification ranking was employed in their study.



(a) RFT GoogleNet Evaluation

(b) RS GoogleNet Evaluation

Figure 28: Reduced GoogleNet test accuracies for different color variants of the MPEG-7 dataset after 10K iterations for RFT-GoogleNet and 30K iterations for RS-GoogleNet.

Table 21: Five-fold cross-validation and test performances (95% Confidence Intervals) on the MPEG-7 Dataset with respect to Top 1 and Top 5 Accuracies

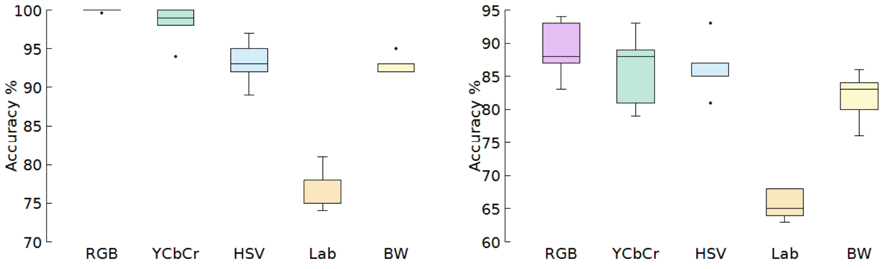
Evaluation Type	CNN Methods	RGB	YCbCr	HSV	Lab	BW
Test (Top 1)	RFT-GoogleNet	90.71 ± 0.45	**91.94 ± 0.61**	**91.68 ± 1.32**	89.82 ± 1.46	91.23 ± 1.03
	RS-GoogleNet	**89.37 ± 0.70**	89.20 ± 1.17	**89.74 ± 0.79**	89.31 ± 1.02	88.85 ± 1.17
Validation (Top 1)	RFT-GoogleNet	90.00 ± 2.59	92.14 ± 1.58	90.86 ± 2.39	89.85 ± 2.00	91.42 ± 1.90
	RS-GoogleNet	87.71 ± 2.52	88.57 ± 2.38	89.42 ± 2.97	87.71 ± 2.95	88.29 ± 2.70
Test (Top 5)	RFT-GoogleNet	96.20 ± 0.45	**97.34 ± 0.30**	**97.43 ± 0.64**	96.51 ± 0.82	96.83 ± 0.44
	RS-GoogleNet	**96.49 ± 0.22**	**96.40 ± 0.50**	**96.46 ± 0.55**	96.31 ± 0.42	96.29 ± 0.53
Validation (Top 5)	RFT-GoogleNet	96.43 ± 0.97	97.29 ± 0.92	97.43 ± 0.75	96.57 ± 1.28	96.86 ± 0.64
	RS-GoogleNet	96.71 ± 2.15	96.71 ± 1.29	96.29 ± 2.00	96.29 ± 2.29	96.29 ± 1.34

Table 22: Comparison of our approaches to other methods on the MPEG-7 dataset.

Methods	Test Accuracy
Contour Segment and Skeleton Path (Bai et al., 2009)	96.60
Region Area Descriptor-Simplified Shape Signature-RSD (RAD+SSS+RSD) (Lin et al., 2017)	91.43
Contour Segment (Bai et al., 2009)	91.10
Segment Set (Sun and Super, 2005)	90.90
Region Area Descriptor-Simplified Shape Signature (RAD+SSS) (Lin et al., 2017)	89.57
Skeleton Path (Bai et al., 2009)	86.70
RFT-GoogleNet-YCbCr	**91.94**
RFT-GoogleNet-HSV	**91.68**

5.3.3 Evaluation on the Wild-Anim Dataset

We employed the train-validation sets to create models during training, which are used to evaluate the various color variants of this dataset, using reduced instances of the already described CNNs. We computed the mean test distributions (with 95% confidence intervals) after 10K iterations for RFT-GoogleNet and 30K iterations for RS-GoogleNet as shown in Figure 29. The results obtained are also shown in Table 23. The results show that RFT-GoogleNet applied on the YCbCr version of this dataset obtained an accuracy of 98.2% which outperforms most of the other color versions of this dataset, except for the state-of-the-art (Okafor et al., 2016) technique using RFT-GoogleNet applied on the RGB version of the dataset. This is expected because the fine-tuned filters from GoogleNet were trained on natural images which have particular color patterns. Because the pre-trained net was trained on RGB images, it also learned to extract features from the patterns existing in RGB space. It can be observed from Table 23 that GoogleNet with both scratch and fine-tuned versions applied on YCbCr and HSV significantly outperform the use of the Lab color space. Still, also for scratch GoogleNet, the use of the RGB images leads to the best results, which shows that natural images are best described using this color space. Please note that the reduced instances of GoogleNet applied on most of the color spaces $\{RGB, YCbCr, HSV\}$ outperform the use of the BW version of this dataset. This shows the importance of color information in deep learning. However, we observed that the CNN on the Lab color space yields the worst performance on this dataset, which can be explained by the fact that the transformation from the natural image (RGB) color space to the Lab color space resulted in the loss of useful pixel information as shown in Figure 22. That is not the case in the transformation which created the Lab versions from the binary masked images as discussed in the previous subsections, where the CNN trained on the Lab color space yielded fairly good results compared to the CNN trained on the other color spaces.



(a) RFT GoogleNet Evaluation (b) RS GoogleNet Evaluation

Figure 29: Reduced GoogleNet test accuracies for different color variants of the Wild-Anim dataset after 10K iterations for RFT-GoogleNet and 30K iterations for RS-GoogleNet.

Table 23: Average test accuracies (with 95% Confidence Intervals) on the Wild-Anim Dataset

CNN Methods	RGB	YCbCr	HSV	Lab	BW
RFT-GoogleNet	**99.93** (Okafor et al., 2016)	**98.20 ± 1.95**	93.20 ± 2.38	77.20 ± 2.18	93.00 ± 0.96
RS-GoogleNet	**89.00** (Okafor et al., 2016)	**86.00 ± 4.57**	**86.60 ± 3.39**	65.60 ± 1.89	81.80 ± 3.06

5.3.4 Significance Test

We employed student’s T-tests to determine the significance of the results reported in the previous subsections.

5.3.4.1 *Animal-Shape Dataset*

The results show that the CNNs trained on some of the color variant images yield marginal performance gains compared to the CNN trained on the original BW images of the dataset. In general there is no significant difference using CNN applied to either color variants or BW on this dataset.

5.3.4.2 *MPEG-7 Dataset*

In subsection 5.3.2, we observed a marginal performance gain training CNNs on some of the color variants of this dataset when compared to CNNs that are trained on the original images (BW). However, the CNNs trained on the color variants show no significant difference ($p > 0.05$) when compared to the CNNs trained on the original images. This can be explained by the fact that the CNNs, when trained on either the color or the BW versions of this dataset, have already attained a very good level of performance (even better than for the Animal-Shape dataset despite consisting of more classes).

5.3.4.3 *Wild-Anim Dataset*

The use of finetuned CNNs trained on RGB and YCbCr images yield significantly higher classification accuracies ($p < 0.05$) when compared to the CNN trained on the BW images of this dataset.

5.4 Conclusion

We developed a color space conversion algorithm that has the potential to transform images to other color variants of image datasets. Then, the datasets were fed as input to two broad customized versions of the GoogleNet architecture. The algorithm provides informative pixels to the black and white binary masked images, which is especially clear in the datasets which are transformed to YCbCr color space.

We remark that this is the first time an evaluation is examined on different color versions of different shape and natural image datasets. The evaluation is performed on three datasets using some instances of

GoogleNet. The results on the MPEG-7 dataset show that our proposed method performs very well. Currently our best Top-1 result happens to be the second best approach reported on this dataset. Furthermore, the results show that the reduced finetuned versions of GoogleNet trained on the new color versions or BW images of the Animal-Shape dataset outperform all existing methods. We also remark that for classifying natural images, the use of the RGB color space leads to the best recognition accuracies.

Most of the CNNs combined with either YCbCr and HSV color spaces yielded marginal performance gains relative to CNNs trained on the binary masked (BW) and Lab color space for most of the three dataset examined. This research creates a new exploration in the use of either YCbCr or HSV for deep colorization since they may compute useful pixel intensities for the input layer of the CNNs that could aid to boost recognition accuracies.

Further work could involve the use of training residual or highway networks on color variants of the MPEG-7 and Animal-Shape datasets. The mentioned architectures may improve the current performances on different color variants of a given BW image dataset. We also want to study the application of other forms of data augmentation techniques (such as color casting, elastic deformation, rotation, and scaling) to enhance performances on different color variants of a given dataset. Finally, we want to explore the possibility of using generative adversarial networks for artificial colorization of images using YCbCr or HSV color spaces.

6 DETECTION AND RECOGNITION OF BADGERS USING DEEP LEARNING

This chapter describes the use of two different deep-learning algorithms for object detection to recognize different badgers. We use recordings of four different badgers under varying background illuminations. In total four different object detection algorithms based on deep neural networks are compared: The single shot multi-box detector (SSD) with the Inception-V2 or MobileNet as a backbone, and the faster region-based convolutional neural network (Faster R-CNN) combined with Inception-V2 or residual networks. Furthermore, two different activation functions are compared to compute probabilities that some badger is in the detected region: the softmax and sigmoid functions. The results of all eight models show that SSD obtains higher recognition accuracies (97.8% - 98.6%) than Faster R-CNN (84.8% - 91.7%). However, the training time of Faster R-CNN is much shorter than that of SSD. The use of different output activation functions seems not to matter much.

This chapter is published in:

Okafor, E., Berendsen, G., Schomaker L.R.B., and Wiering, M.A. (2018). Detection and Recognition of Badgers using Deep Learning. *The International Conference on Artificial Neural Networks (ICANN), Lecture Notes in Computer Science book series; vol. 11141, pages 554-563. Springer, Cham.*

Badgers are short-legged omnivores and wild animals, and their existence is in danger in some parts of the world. To control this threat, some countries in Europe: United Kingdom, France, Republic of Ireland, Northern Ireland, and the Netherlands formed the Eurobadger collaboration with the objective to protect the existence of badgers. To assist this protection, there is a need to deploy computer vision systems that can aid in detecting and recognizing these animals, whose habitat is often a network of underground tunnels (setts). This chapter describes the use of several deep neural network approaches to detect and classify different badgers.

Previous research (Koik and Ibrahim, 2012) suggests that the human eye is an efficient and reliable method for animal detection. However, the effectiveness of the human eye reduces due to tiredness and a human is not able to focus on an animal for 24 hours a day. Therefore it is more efficient to apply computer-vision techniques for detecting and recognizing animals. Early research in (Burghardt and Calic, 2006) detects animal faces using Haar-like features and the Adaboost classifier, while tracking the animals was done using the Kanade-Lucas-Tomasi method. Researchers have investigated different approaches to detect animals or humans: detection of humans in motion using background subtraction (BG) (Chen, 2009), using frame differences with the $W4$ algorithm (Sengar and Mukhopadhyay, 2017), using background frame differences based on Gaussian functions (Liu and Hou, 2012), and the combination of BG and three-frame differencing (Liu et al., 2016a).

Since the emergence of deep neural networks in the computer vision community, they have gained a lot of attention and successes for solving different learning tasks such as classification of objects, plants, and animals (Krizhevsky et al., 2012), (Szegedy et al., 2015), (He et al., 2016a), classifying wild-animals (Okafor et al., 2016), and recognizing cows with unmanned aerial vehicles (UAVs) using data-augmented images (Okafor et al., 2017, 2018). Concerning wildlife monitoring and conservation, the authors in (Christiansen et al., 2017) investigated an automated detection and classification method of animals or non-animals using thermal images. Their method is based on the discrete cosine transform for feature extraction and k-nearest neighbors for classification. The research in (Gonzalez et al., 2016) approaches wildlife monitoring using UAVs that use thermal image acquisition and a video processing pipeline to provide automatic detection, classification, and tracking of wildlife in a forest or open area. Recent research in (He et al., 2016c) unites some scientists with the objective of monitoring wildlife. Their study showed that convolutional neural networks outperform a more classical technique based on the bag of visual words with a support vector machine in their wildlife detection challenge.

To the best of our knowledge, no research has been done concerning the detection and recognition of different badgers. The challenge is that some of the examined badgers have very similar color appearances, and therefore accurately discriminating the various badgers could be a difficult problem for computer vision algorithms.

Contributions: This chapter proposes the use of several object detection algorithms based on deep neural networks for detecting and recognizing badgers from video data. For this, a comparison is made between two neural network based detectors: SSD (Liu et al., 2016b) and Faster R-CNN (Ren et al., 2015). SSD is combined with the Inception-V2 (Ioffe and Szegedy, 2015) or MobileNet (Howard et al., 2017) as a backbone and the Faster R-CNN detector is combined with either Inception-V2 or Residual networks (He et al., 2016a) with 50 layers (ResNet-50) as feature extractors. Furthermore, we compare the use of two output activation functions: the softmax and sigmoid function. For the experiments, we use several videos recorded with a low-resolution camera. The results show that most of the trained SSD detectors significantly outperform the different variants of the Faster R-CNN detector. All the Faster R-CNN methods are computationally much faster than the SSD techniques for training the system, although for testing SSD is a bit faster.

Outline: Section 6.1 describes the dataset used and the preprocessing steps. Section 6.2 explains the detection algorithms and the experimental setup for training the models. Section 6.3 presents the results. Section 6.4 concludes the paper and provides directions for future research.

6.1 Dataset and Preprocessing

The dataset is based on videos of different badgers collected by the foundation of Das & Boom[1]. The dataset contains four individual instances of

[1] http://www.dasenboom.nl/

badgers with a total number of 51 videos. The badger classes (identities) are: *badger_esp*, *badger_iaco*, *badger_looi*, and *badger_strik*. The badgers were recorded in 2016 and 2017 at the Badger Rescue Center of Das & Boom in the Netherlands. Additionally some videos and photos were made at release locations for badger rehabilitation purposes. To identify each badger, they are micro-chipped so the animal can be tracked during captivity and identified after release.

The streaming lengths (T_s) of the videos vary in the range between 15 and 60 seconds. We extracted approximately a frame per second, for which we developed a script that extracts $(Ts \pm 2)$ video frames. We remark that some frames do not contain the existence of badgers and such frames are not used in our experiments. The details of the used dataset are shown in Table 24. Some example images of the used dataset are shown in Figure 30.

Table 24: Dataset description

Badger Class	No. of Videos	Dataset-split (frames)		$T_s(s)$
		Train	Test	
Badger_esp	7	328	28	59
Badger_iaco	28	323	30	15
Badger_looi	9	437	61	59
Badger_strik	7	372	62	60
Total	51	1460	181	

We now describe how we made the ground truth annotations for the detection task. We used one video to create the images for the test set for each of the classes except for *Badger_iaco* where two videos are used in the test set. The remaining videos are used to create the train set. Manual extraction of the bounding box containing the existence of a badger was done using the LabelImg[2] tool. The used tool provides the annotation of a given image, and it is saved in the *.xml* file format. Each of the annotation files contains 4 coordinates representing the location of the bounding box

[2] //github.com/tzutalin/labelImg

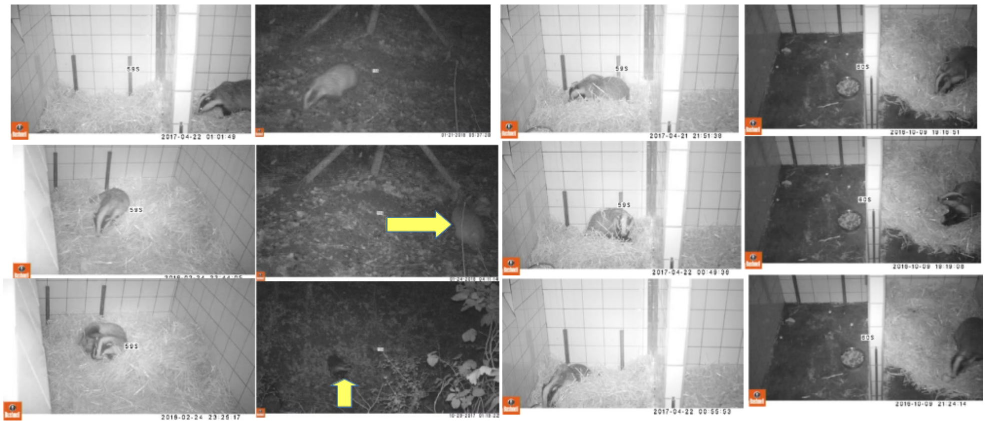


Figure 30: Example images present in the Badger dataset; where each column represents: *Badger_esp*, *Badger_iaco*, *Badger_looi*, and *Badger_strik* respectively. The yellow arrows in column two indicate the existence of *badger_iaco* under poor illumination conditions (environment). Note that most videos were shot while the badgers were in captivity for a while, although some videos were shot in the wild.

surrounding the badger, the label and the file path to the images. We employed the Pascal VOC format.

6.2 Methods

This section describes the used deep neural network detection frameworks. Figure 31 shows the overall network pipeline that consists of data preprocessing as presented in Section 6.1, training the CNN to obtain the different detection models and their corresponding real-time deployment.

6.2.1 SSD with Inception-V2

The Single Shot multi-box-Detector (SSD) (Liu et al., 2016b) is a detection framework that employs feed-forward convolutional neural networks for prediction of object classes and anchor offsets, with no consideration for second phase classification. Instead, it uses non-maximum suppression that allows the final detection of the objects in a single pass. A unique characteristic of this framework is that multi-scale convolutional bounding

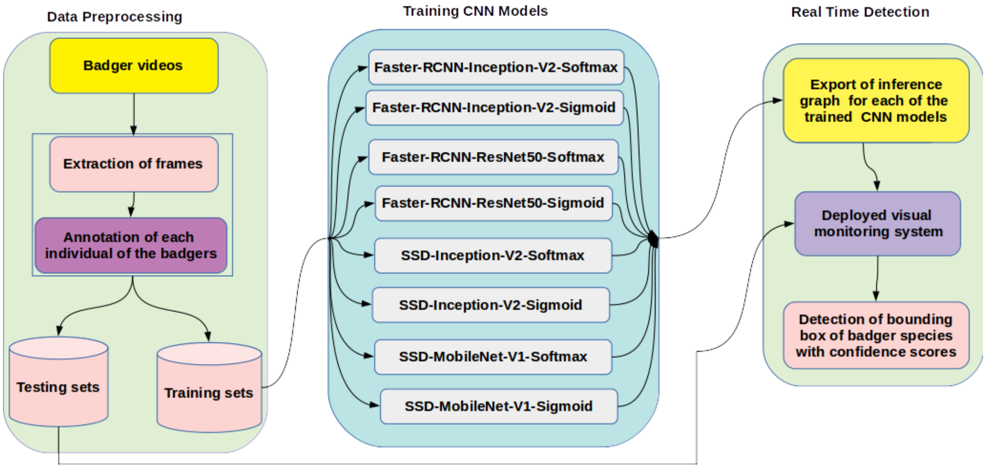

Data Preprocessing
Badger videos
Extraction of frames
Annotation of each individual of the badgers
Testing sets
Training sets
Training CNN Models
Faster-RCNN-Inception-V2-Softmax
Faster-RCNN-Inception-V2-Sigmoid
Faster-RCNN-ResNet50-Softmax
Faster-RCNN-ResNet50-Sigmoid
SSD-Inception-V2-Softmax
SSD-Inception-V2-Sigmoid
SSD-MobileNet-V1-Softmax
SSD-MobileNet-V1-Sigmoid
Real Time Detection
Export of inference graph for each of the trained CNN models
Deployed visual monitoring system
Detection of bounding box of badger species with confidence scores

Figure 31: Overall pipeline for the real-time detection systems; the first box accounts for the data-preprocessing, the second box represents the training of the CNN detection system, and the last box provides the network the inference generator and visual monitoring deployment system in the testing phase.

box outputs are attached to several feature maps at the top of the network layer. At the bottom or base portion of the network, the feature extraction method Inception-V2 (Ioffe and Szegedy, 2015) increases the breadth and depth of the network with a quite low computational complexity due to the used inception modules. The Inception-V2 extracts feature maps from the input images. The combination of SSD and Inception-V2 is called SSD-Inception-V2 (Maeda et al., 2018). We examine two forms of classification activation functions; sigmoid and softmax. This results in two variants of this approach.

Network Setup: we have trained the network using pre-trained weights *ssd_inception_v*2 _*coco*_2017_11_17, originally trained by a group of Google researchers. The pre-trained weights contain information from a subset of the Microsoft common object in context (COCO) dataset (Lin et al., 2014) containing a total of approximately 328K images with different object classes. We further trained the network using badger images with bounding boxes and class labels as input to the training algorithm. This use of pre-trained weights has the benefit of less training time compared to training random weights from scratch that demands longer computing times. During training of the network, we adopted a similar experimental setup as in (Liu et al., 2016b) because it yields good performances. The network parameters include; the original input image frames contain 427 × 240 pixels and are resized online to 300 × 300 pixels, the convolutional box predictor uses a prediction dropout probability 0.8, kernel size 3 × 3 and a box-code size set to 4. The root mean square propagation (RMSprop) optimization algorithm is used for optimizing the loss functions trained for 40,000 steps using the following parameters; a learning rate of 0.004, decay factor 0.95, and decays at an interval of 16,000 steps. At the non-maximum suppression part of the network a score threshold of 1×10^{-8} is used with an intersection of union (IoU) threshold of 0.6, both the classification and localization weights are set to 1.

6.2.2 SSD with MobileNet-V1

This method also uses SSD (Liu et al., 2016b) for detection while the MobileNet-V1 (Howard et al., 2017) as the base network is used as feature extractor. A MobileNet is a neural network based feature extractor that employs depth-wise separable filters for extracting feature maps from a given image. The depth-wise separable convolution in this network involves the integration of depth-wise convolution and 1 × 1 point-wise convolution. The merit of this approach is that it reduces computational cost compared to standard convolution (Howard et al., 2017). The output from the MobileNet is further processed using SSD. The method is referred to as SSD-MobileNet-V1. Additionally, we consider two forms of classification activation functions: sigmoid and softmax. This results in two variants of this method.

Network Setup: we have trained the network using pre-trained weights *ssd_mobilenet_v*1 _*coco*_2017_11_17 from the COCO dataset as was explained in the previous subsection, for training our custom network using the badger images as input to the SSD-MobileNet-V1 system. The training process uses similar hyperparameters as described in the previous subsection.

6.2.3 Faster R-CNN with ResNet-50

The Faster R-CNN algorithm (Ren et al., 2015) is an improvement of Fast R-CNN (Girshick, 2015). In this network, the working operation of the Faster R-CNN involves two phases. The first phase requires the use of a region proposal network (RPN) which allows concurrent prediction of object anchors and confidence (objectiveness) from some intermediate layers. Note that a feature extraction network can be used for this purpose, in this case, a residual network with a depth of 50 layers (ResNet-50) (He et al., 2016a) is used. The second phase requires information from the first phase to make an accurate prediction of the class label and its bounding box refinement. Additionally, we made consideration of the classification activation functions that were earlier discussed in the previous subsections. Hence this results in two variants of this network.

Network Setup: we have trained the network using pre-trained weights $faster\_rcnn\_resnet50\ \_coco\_2018\_01\_28$ from the COCO dataset. The training of the network factored in some modified experimental setups as in (Ren et al., 2015). The original input image (badger) to the network contains 427×240 pixels and is resized online with an aspect ratio of min-max dimensions $[600, 1024]$ during training. As earlier discussed the network comprises of two phases. The first phase initiates a grid-anchor of size 16×16 pixels with scales $[0.25, 0.5, 1.0, 2.0]$, a non-maximum-suppression-IoU-threshold set to 0.7, the localization loss weight 2.0, objectiveness weight 1.0 with an initial crop size of 14×14 pixels, kernel size 2×2 with strides set to 2. The second phase computes the prediction score with IoU-threshold set to 0.6; the SGD optimizer optimizes the loss functions using an initial learning rate 0.0002 and momentum value 0.9. Again, the network was trained for 40,000 steps.

6.2.4 Faster R-CNN with Inception-V2

The Faster R-CNN detector employs an Inception V2 feature extractor for extracting useful feature maps from an input image. The intermediate layer from the Inception module uses the RPN component of the network for prediction of object anchors and confidences. Similar procedures as explained in (Ren et al., 2015) were followed.

Network Setup: we have trained the network using pre-trained weights $faster\_rcnn\_inception\ \_v2\_coco\_2018\_01\_28$ from the COCO dataset. The training of our custom network employs the badger images as input to the Faster-RCNN-Inception-V2 system. The training process uses similar hyperparameters as described in the previous subsection.

All the experiments were carried out using the Tensorflow object detection API framework on a Ge-Force GTX 960 GPU model, and the operating system platform employed is Ubuntu 16.0. We modified the deployment script in the Tensorflow object detection API, by providing the possibility to evaluate all images in the test directory instead of applying restrictions. Moreover, we also use our own script to compute the performance index metrics of the used methods. The next section discusses the performance and overall training time for each of the methods.

6.3 Results

The overall training time for each of the used methods is reported in Table 25. The table shows that the training time of Faster R-CNN is much shorter than the training time of SSD, and the use of Inception-V2 leads to the shortest training times. The frame rates show that most of the methods can analyze 0.8 - 1.5 images per second using our hardware, and SSD is a bit faster than Faster R-CNN for deployment.

We carried out two experimental runs and computed the average precision, recall and accuracy, based on the predicted class label in a detected box. The standard deviations for all the methods are $\leq 1.4\%$, which indicates that the performances of the techniques are consistent.

The summary of the average performance indices and the standard deviations for each of the methods is presented in Table 26. From this table, we draw the conclusion that SSD-Inception-V2 for both output functions and the SSD-MobileNet-V1-Sigmoid outperforms all the Faster R-CNN variants with $p < 0.05$ significance level.

The performance index from the SSD-network variants provides a more precise detection compared to the Faster R-CNN network variants. The

Table 25: Average time evaluation for the different detection systems

Methods (CNN Models)	Training Time	Time Improvement	Testing Time	Frame Rate (f/s)
Faster_RCNN-Inception_V2_Sigmoid	$3h, 21m$	×3.0	222s	0.82
Faster_RCNN-Inception_V2_Softmax	$3h, 23m$	×2.9	211s	0.86
Faster_RCNN-ResNet-50-Sigmoid	$5h, 37m$	×1.4	268s	0.68
Faster_RCNN-ResNet-50-Softmax	$5h, 44m$	×1.3	267s	0.68
SSD_Inception_V2_Softmax	$10h, 45m$	×0.24	162s	1.12
SSD_Inception_V2_Sigmoid	$10h, 46m$	×0.24	163s	1.11
SSD_MobileNet_V1_Softmax	$13h, 16m$	×0.01	120s	1.51
SSD_MobileNet_V1_Sigmoid (Baseline)	$13h, 21m$	$---$	122s	1.48

lower precision in the Faster R-CNN may have arisen due to localization bias problems. Figure 32 shows some examples of the detection scores of badgers within a given image during testing evaluation. From this figure, we observe that the Faster R-CNN methods misclassified this particular example of badger_strik (gray box) as badger_esp (green box) as shown in sub-images within cells (3,2) and (4,2). Hence, this explains the lower performance index using the Faster R-CNN methods compared to the SSD network variants. From an application standpoint, it could be profitable to use Inception-V2 as the backbone for the SSD detector, since it presents more precise detections of the objects of interest. Additionally, the results suggest that SSD-based networks are useful in handling localization bias problems.

Table 26: Average performances for the different detection and recognition systems

Methods (CNN Models)	Performance Index		
	Precision	Recall	Accuracy
SSD_Inception_V2_Softmax	0.988 ± 0.012	0.986 ± 0.014	0.986 ± 0.014
SSD_Inception_V2_Sigmoid	0.986 ± 0.003	0.986 ± 0.004	0.986 ± 0.004
SSD_MobileNet_V1_Sigmoid	0.985 ± 0.005	0.983 ± 0.006	0.983 ± 0.006
SSD_MobileNet_V1_Softmax	0.978 ± 0.011	0.978 ± 0.011	0.978 ± 0.011
Faster_RCNN-Inception_V2_Softmax	0.942 ± 0.009	0.917 ± 0.011	0.917 ± 0.011
Faster_RCNN-Inception_V2_Sigmoid	0.945 ± 0.003	0.914 ± 0.008	0.914 ± 0.008
Faster_RCNN-ResNet-50-Sigmoid	0.936 ± 0.000	0.890 ± 0.000	0.890 ± 0.000
Faster_RCNN-ResNet-50-Softmax	0.921 ± 0.003	0.848 ± 0.003	0.848 ± 0.003

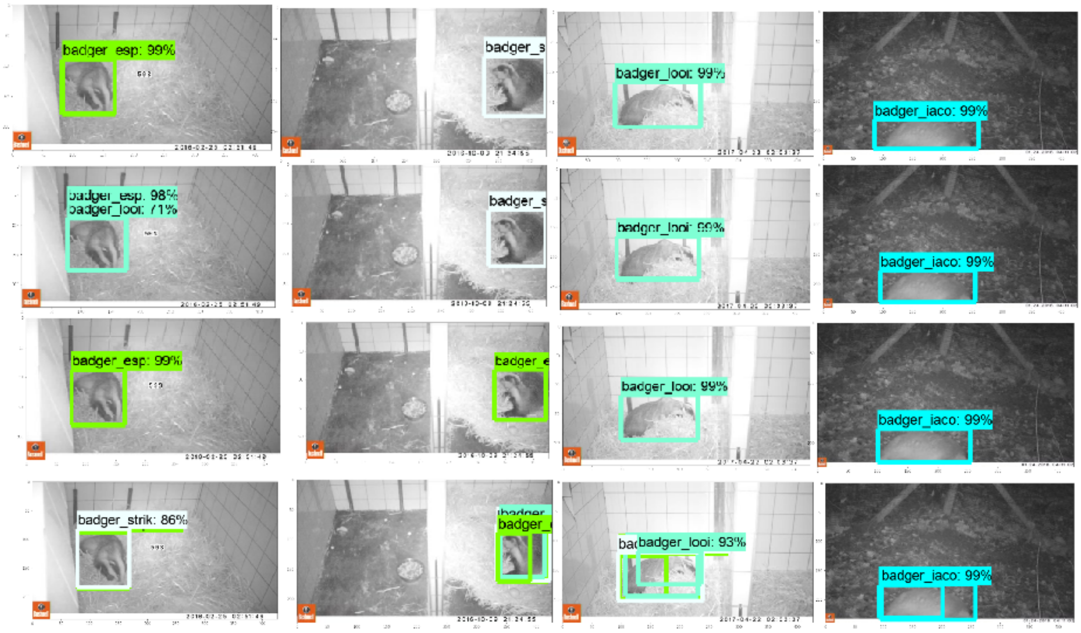


Figure 32: Testing detection confidence prediction of the badger individual instances using different neural networks detection methods: the first row indicates detection using ssd_mobilenet_v1_softmax, the second row shows the detection using ssd_inception_v2_sigmoid, the third row shows the detection using faster_rcnn_inception_v2_softmax, and the last row shows the detection using faster_rcnn_resnet50_sigmoid. Note that each of the columns represents the badger individual instances in the order; *Badger_esp*, *Badger_strik*, *Badger_looi*, and *Badger_iaco* respectively.

6.4 Remarks

Real-time detection using deep learning can be used for many localization and identification tasks. In this chapter, several deep neural networks were used to detect and classify different badgers using a novel animal dataset. We compared the single shot multi-box detector (SSD) combined with Inception-V2 or MobileNet, to faster-region-based convolutional neural network (Faster R-CNN) combined with Inception-V2 or residual networks (ResNet). We used the pre-trained networks and further trained them on our dataset. The four detectors were combined with either a softmax or sigmoid function for computing the output probability scores, hence resulting in eight different models.

The results showed that SSD with the Inception-V2 as a backbone obtains the highest mean accuracy performance (98.6%). Furthermore, we noticed that during testing, SSD has a higher frame rate than Faster R-CNN, although its training time is longer. Our analyses suggests that the examined SSD methods tackle the problem of localization bias much better than Faster R-CNN during prediction of the bounding boxes. Finally, we noticed that the use of the sigmoid or softmax output activation functions led to comparable results.

Future work will be directed at the scalability in the number of classes and environments, using a much larger dataset. We also suggest that the best found model, SSD-Inception-V2-Softmax, could be improved and deployed into UAVs or thermal acquisition cameras, as this can help to detect badgers in environments where they are endangered.

DISCUSSION 7

Animal recognition and detection have received research attention due to the need to: estimate the animal population size, protect endangered instances of animals, allow governments of some countries to make conservation and monitoring-policy for wildlife. This dissertation has examined different computer vision techniques, proposing compact deep neural network methods as feature extractors combined with supervised learning algorithms to perform tasks such as animal recognition, detection or both. This thesis focused on three main problems; the relevance of neural network reduction, enhancement of images using novel approaches to data augmentation or artificial colorization and the use of deep neural networks for a real-time detection system.

The remaining components of this chapter provides solutions to the three main problems and answers to the research questions. Section 7.1 highlights the recommendations for future work.

Comparison of classical methods to customized deep learning methods

In Chapter 2, we have analyzed the best image recognition technique when there are not many images. For this, the classical computer vision systems were compared to deep learning systems for recognizing animal images to answer the research question. The next research question assessed what benefit does reducing the number of neurons from existing deep learning architectures present. To answer this question, we proposed effectively reduced neural network architectures with fewer network parameters during the training of the recognition systems. One merit of the proposed approaches is that it lowers computational costs for some approaches and yields almost similar levels of performance when compared to the existing deep learning techniques. The use of deep learning methods significantly surpasses most of the classical computer vision methods when examined on our relatively small dataset.

Rotation-matrix data augmentation on UAV images

In Chapter 3, we introduced a novel algorithm that handles rotation variation from unmanned aerial vehicle images without creating too many images. To answer the research question, we proposed a rotation matrix algorithm that transforms an image to a new image containing randomly rotated copies of sub-images defined within a given grid of cells and angular bounds. The advantage of the proposed algorithm (data augmentation method) is that it enhances the pixel information in an image. It does not require an increase in the number of images compared to conventional data augmentation methods during training of the recognition systems.

We investigated another research question on how well the shallow depth neural network systems and classical computer vision techniques combined with different supervised learning algorithms can deal with recognizing several kinds of images: rotation-matrix data augmentation and original images. To answer the research question, we used several instances of scratch and finetuned version of a customized GoogleNet architecture with only three inception modules while considering two different loss functions: cross-entropy loss and hinge-loss. The resulting deep learning methods were compared to classical computer vision methods such as a local descriptor (histogram of oriented gradients), a global descriptor (color histogram) and the combination of both. These descriptors are used for creating feature activations, and the resulting feature vectors from each of the descriptors are passed to the supervised learning algorithm such as radial basis and linear support vector machines, and K-nearest neighbor.

The results suggested that the fine-tuned versions of our customized recognition system when combined with the rotation matrix data augmentation images yielded the best performance when compared to all other approaches.

Unification of rotation matrix and color constancy

In Chapter 4, we have developed a recognition system that is robust to illumination variation and determined if the integration of the rotation matrix and color constancy images can aid to yield a robust recognition system. To answer the research question, we proposed the combination of both color-constancy and rotation matrix data augmentation algorithms for transforming an input image. The merit of the proposed data augmentation method includes; it increase the number of training images especially for those datasets with a relatively small amount of images. Furthermore, it enhances the illumination quality of a blurred image.

Additionally, we determined what relevance the choice of grid size and angular bound selections have for the proposed data augmentation method. To answer the research question, we have investigated the recognition system while considering different grid sizes and angular bound conditions on three animal datasets. The results suggested that appropriate choices of grid size and angular bounds can help the recognition system to obtain robust classification accuracies.

Analysis of color spaces for image recognition in deep learning

In Chapter 5, we have analyzed the importance of the use of color spaces in deep learning methods. What is the performance of neural network systems for different color space variants for the examined animal image datasets? To answer the research question, we constructed a color conversion algorithm that has the prowess to transform natural (RGB) or black and white (BW) images to four other color spaces. Then we employed our custom network systems to access the classification performance on several variants of the used animal datasets.

The merit of our proposed artificial colorization algorithm combined with the customized recognition systems yielded marginal performance gains when compared to the original image (BW) on most of the used datasets. Moreover, the results suggested the importance of training a recognition system on natural images to obtain the best performance.

Detection and recognition of badgers using deep learning

In Chapter 6, we analyzed detection algorithms for detecting and recognizing individual instances of badgers. The problem here is that localization and finding an object of interest in an image is very difficult for classical computer vision approaches, mainly when there exist high similarities in object appearances.

Our research question is to determine which of the detection neural network systems is the most suitable for application purposes, especially in the deployment phase. To answer the research question, we have investigated several neural network based detection systems for detecting and recognizing individual instances of badgers from video data. We considered two detection algorithms, each combined with some network back-bones: single shot multi-box detector combined separately to Inception V2 and MobileNet-V1, and faster region-based convolutional neural network combined separately to Inception-V2 or ResNet with 50 layers depth. The results suggested that the single shot multi-box detector detectors significantly outperform the faster region-based convolutional neural network detectors.

However, all the faster region-based convolutional neural network methods are computationally much faster than the single shot multi-box detector techniques for training the system, although for testing the single shot multi-box detector is a bit faster. We suggest that the best-found model, single shot multi-box detector combined with Inception-V2 presented better handling of the localization bias problem when compared to other approaches. We recommend that the mentioned best model can be deployed into real-time systems such as a drone or other data-acquisition systems.

7.1 Future work

This dissertation has demonstrated that deep neural networks combined with novel approaches to data augmentation and artificial colorization often improve animal recognition compared to traditional methods.

To further improve the compacted neural network system, there is a need to optimize the filter sizes and hyperparameters using optimization algorithms such as genetic algorithms or surrogate-based optimization algorithms, as the proposed approach may improve recognition accuracies. In the area of image enhancement, future work can develop a neural network layer that has the prowess to learn the proposed rotation matrix algorithm or color-constancy variant as well. This recommended network layer could be combined with other data-augmentation methods to combat overfitting. Additionally, more work can be directed to this research question: will an end-to-end convolutional neural network be able to learn all transformations or is well thought human design work needed? Another interesting research that can be investigated is the implementation of a neural network system with the ability to optimize the angular bounds or grid selection for a given image dataset, to obtain highest recognition accuracy.

Furthermore, more research work can be done to improve the detection accuracy and computational cost by developing a novel detection algorithm that can be compared to the state-of-the-art method (YOLO-v3). Future work can also investigate improving on the real-time detection system, by examining automatic annotation of the region of interest. Further improvements can be made to the neural network system to factor dataset scalability and more environmental diversities. Finally, it would be interesting to use the neural network system to segment contours of a given object of interest on our novel badger dataset.

Although the recognition or detection system results on the different animal datasets show very high accuracies, in future work more images of different animals need to be collected. This would allow computer vision techniques based on deep neural networks to surpass human experts in recognizing and detecting all kinds of animals.

BIBLIOGRAPHY

Abdullah, A., Veltkamp, R. C., and Wiering, M. A. (2010). Ensembles of novel visual keywords descriptors for image categorization. In *Control Automation Robotics Vision (ICARCV), 11th International Conference on*, pages 1206–1211. IEEE.

Arora, S., Bhaskara, A., Ge, R., and Ma, T. (2014). Provable bounds for learning some deep representations. In *International Conference on Machine Learning*, pages 584–592.

Arróspide, J., Salgado, L., and Camplani, M. (2013). Image-based on-road vehicle detection using cost-effective histograms of oriented gradients. *Journal of Visual Communication and Image Representation*, 24(7):1182–1190.

Bai, X., Liu, W., and Tu, Z. (2009). Integrating contour and skeleton for shape classification. In *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, pages 360–367. IEEE.

Bensaali, F. and Amira, A. (2004). Design and efficient FPGA implementation of an RGB to YCrCb color space converter using distributed arithmetic. In *International Conference on Field Programmable Logic and Applications*, pages 991–995. Springer.

Burghardt, T. and Calic, J. (2006). Real-time face detection and tracking of animals. In *Neural Network Applications in Electrical Engineering, 2006. NEUREL 2006. 8th Seminar on*, pages 27–32. IEEE.

Caicedo, J. C. and Lazebnik, S. (2015). Active object localization with deep reinforcement learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2488–2496.

Charalambous, C. C. and Bharath, A. A. (2016). A data augmentation methodology for training machine/deep learning gait recognition algorithms. *arXiv preprint arXiv:1610.07570*.

Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. (2018). Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected conditional random fields (crfs). *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848.

Chen, P. (2009). Moving object detection based on background extraction. In *Computer Network and Multimedia Technology, 2009. CNMT 2009. International Symposium on*, pages 1–4. IEEE.

Chen, W., Shi, Y. Q., and Xuan, G. (2007). Identifying computer graphics using HSV color model and statistical moments of characteristic functions. In *IEEE International Conference on Multimedia and Expo*, pages 1123–1126.

Cheng, Z., Yang, Q., and Sheng, B. (2015). Deep colorization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 415–423.

Christiansen, P., Kragh, M., Steen, K. A., Karstoft, H., and Jørgensen, R. N. (2017). Platform for evaluating sensors and human detection in autonomous mowing operations. *Precision Agriculture*, 18(3):350–365.

Ciresan, D., Meier, U., Gambardella, L., and Schmidhuber, J. (2011). Convolutional neural network committees for handwritten character classification. In *Document Analysis and Recognition (ICDAR), 11th International Conference on*, pages 1135–1139.

Coates, A., Carpenter, B., Case, C., Satheesh, S., Suresh, B., Wang, T., Wu, D. J., and Ng, A. Y. (2011a). Text detection and character recognition in scene images with unsupervised feature learning. In *Document Analysis and Recognition (ICDAR), International Conference on*, pages 440–445. IEEE.

Coates, A., Ng, A. Y., and Lee, H. (2011b). An analysis of single-layer networks in unsupervised feature learning. In *International conference on artificial intelligence and statistics*, pages 215–223.

Csurka, G., Dance, C. R., Fan, L., Willamowski, J., and Bray, C. (2004). Visual categorization with bags of keypoints. In *Computer Vision (ECCV), 8th European Conference on*, pages 1–22.

Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition (CVPR). IEEE Computer Society Conference on*, volume 1, pages 886–893.

Dickinson, P., Qing, C., Lawson, S., Freeman, R., et al. (2010). Automated visual monitoring of nesting seabirds. In *Workshop on visual observation and analysis of animal and insect behaviour, Istanbul.*

Elgammal, A., Harwood, D., and Davis, L. (2000). Non-parametric model for background subtraction. In *European conference on computer vision*, pages 751–767. Springer.

Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., and Lin, C.-J. (2008). Liblinear: A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874.

Fang, Y., Du, S., Abdoola, R., Djouani, K., and Richards, C. (2016). Motion based animal detection in aerial videos. *Procedia Computer Science*, 92:13–17.

Fu, M., Xu, P., Li, X., Liu, Q., Ye, M., and Zhu, C. (2015). Fast crowd density estimation with convolutional neural networks. *Engineering Applications of Artificial Intelligence*, 43:81–88.

Ghazi, M. M., Yanikoglu, B., and Aptoula, E. (2017). Plant identification using deep neural networks via optimization of transfer learning parameters. *Neurocomputing*, 235:228–235.

Girshick, R. (2015). Fast R-CNN. *arXiv preprint arXiv:1504.08083.*

Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Aistats*, volume 9, pages 249–256.

Gong, T., Fan, T., Guo, J., and Cai, Z. (2017). Gpu-based parallel optimization of immune convolutional neural network and embedded system. *Engineering Applications of Artificial Intelligence*, 62:384–395.

Gonzalez, L. F., Montes, G. A., Puig, E., Johnson, S., Mengersen, K., and Gaston, K. J. (2016). Unmanned aerial vehicles (UAVs) and artificial intelligence revolutionizing wildlife monitoring and conservation. *Sensors*, 16(1):97.

Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. (2016). *Deep learning*, volume 1. MIT press Cambridge.

Guilford, T., Meade, J., Willis, J., Phillips, R. A., Boyle, D., Roberts, S., Collett, M., Freeman, R., and Perrins, C. (2009). Migration and stopover in a small pelagic seabird, the manx shearwater puffinus puffinus: insights from machine learning. *Proceedings of the Royal Society of London B: Biological Sciences*.

Gupta, R. K., Chia, A. Y.-S., Rajan, D., Ng, E. S., and Zhiyong, H. (2012). Image colorization using similar images. In *Proceedings of the 20th ACM international conference on Multimedia*, pages 369–378.

He, K., Zhang, X., Ren, S., and Sun, J. (2016a). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

He, K., Zhang, X., Ren, S., and Sun, J. (2016b). Identity mappings in deep residual networks. *arXiv preprint arXiv:1603.05027*.

He, Z., Kays, R., Zhang, Z., Ning, G., Huang, C., Han, T. X., Millspaugh, J., Forrester, T., and McShea, W. (2016c). Visual informatics tools for supporting large-scale collaborative wildlife monitoring with citizen scientists. *IEEE Circuits and Systems Magazine*, 16(1):73–86.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

Hsu, R.-L., Abdel-Mottaleb, M., and Jain, A. K. (2002). Face detection in color images. *IEEE transactions on pattern analysis and machine intelligence*, 24(5):696–706.

Imamura, Y., Okamoto, S., and Lee, J. H. (2016). Human tracking by a multi-rotor drone using hog features and linear svm on images captured by a monocular camera. In *Proceedings of the International MultiConference of Engineers and Computer Scientists*, volume 1, pages 8–13.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Ironi, R., Cohen-Or, D., and Lischinski, D. (2005). Colorization by example. In *Rendering Techniques*, pages 201–210. Citeseer.

Jack, K. (2011). *Video demystified: a handbook for the digital engineer*. Elsevier.

Jaeger, J., Simon, M., Denzler, J., Wolff, V., Fricke-Neuderth, K., and Kruschel, C. (2015). Croatian fish dataset: Fine-grained classification of fish species in their natural habitat. In *British Machine Vision Conference (BMVC)*.

Jeon, S., Shin, J.-W., Lee, Y.-J., Kim, W.-H., Kwon, Y., and Yang, H.-Y. (2017). Empirical study of drone sound detection in real-life environment with deep neural networks. *arXiv preprint arXiv:1701.05779*.

Jiang, B., Woodell, G. A., and Jobson, D. J. (2015). Novel multi-scale retinex with color restoration on graphics processing unit. *Journal of Real-Time Image Processing*, 10(2):239–253.

Jiang, X., Pang, Y., Li, X., and Pan, J. (2016). Speed up deep neural network based pedestrian detection by sharing features across multi-scale models. *Neurocomputing*, 185:163–170.

Jobson, D. J., Rahman, Z.-u., and Woodell, G. A. (1997). A multiscale retinex for bridging the gap between color images and the human observation of scenes. *IEEE Transactions on Image processing*, 6(7):965–976.

Jothi, R., Mohanty, S. K., and Ojha, A. (2015). Fast minimum spanning tree based clustering algorithms on local neighborhood graph. In *Graph-Based Representations in Pattern Recognition*, pages 292–301. Springer.

Junior, O. L., Delgado, D., Gonçalves, V., and Nunes, U. (2009). Trainable classifier-fusion schemes: an application to pedestrian detection. In *Intelligent Transportation Systems*, volume 2.

Karaaba, M. F., Surinta, O., Schomaker, L. R. B., and Wiering, M. A. (2016). Robust face identification with small sample sizes using bag of words and histogram of oriented gradients. In *Proceedings of the 11th Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, pages 582–589.

Katsigiannis, P., Misopolinos, L., Liakopoulos, V., Alexandridis, T. K., and Zalidis, G. (2016). An autonomous multi-sensor UAV system for reduced-input precision agriculture applications. In *Control and Automation (MED), 24th Mediterranean Conference on*, pages 60–64. IEEE.

Khan, R., Van de Weijer, J., Shahbaz Khan, F., Muselet, D., Ducottet, C., and Barat, C. (2013). Discriminative color descriptors. In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*, pages 2866–2873.

Khosla, A., Xiao, J., Torralba, A., and Oliva, A. (2012). Memorability of image regions. In *Advances in Neural Information Processing Systems*, pages 305–313.

Koik, B. T. and Ibrahim, H. (2012). A literature survey on animal detection methods in digital images. *International Journal of Future Computer and Communication*, 1(1):24.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Land, E. H. and McCann, J. J. (1971). Lightness and retinex theory. *Josa*, 61(1):1–11.

Latecki, L. J., Lakamper, R., and Eckhardt, T. (2000). Shape descriptors for non-rigid shapes with a single closed contour. In *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, volume 1, pages 424–429. IEEE.

Lazebnik, S., Schmid, C., and Ponce, J. (2005). A maximum entropy framework for part-based texture and object recognition. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 1, pages 832–838. IEEE.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient based learning applied to document recognition. In *Proceedings of the IEEE*, volume 86(11), pages 2278–2324.

Levin, A., Lischinski, D., and Weiss, Y. (2004). Colorization using optimization. In *ACM Transactions on Graphics (TOG)*, volume 23, pages 689–694.

Li, C., You, X., Ben Hamza, A., Zeng, W., and Zhou, L. (2011). Distinctive parts for shape classification. In *Wavelet Analysis and Pattern Recognition (ICWAPR), 2011 International Conference on*, pages 97–102. IEEE.

Li, Y., Zhu, J., and Li, F. (2010). A hierarchical shape tree for shape classification. In *Image and Vision Computing New Zealand (IVCNZ), 2010 25th International Conference of*, pages 1–6. IEEE.

Li, Z., Liu, J., Tang, J., and Lu, H. (2015). Robust structured subspace learning for data representation. *IEEE transactions on pattern analysis and machine intelligence*, 37(10):2085–2098.

Li, Z. and Tang, J. (2015). Weakly supervised deep metric learning for community-contributed image retrieval. *IEEE Transactions on Multimedia*, 17(11):1989–1999.

Lim, K.-L. and Galoogahi, H. K. (2010). Shape classification using local and global features. In *Image and Video Technology (PSIVT), 2010 Fourth Pacific-Rim Symposium on*, pages 115–120. IEEE.

Lin, C., Pun, C.-M., Vong, C.-M., and Adjeroh, D. (2017). Efficient shape classification using region descriptors. *Multimedia Tools and Applications*, 76(1):83–102.

Lin, T.-Y., Cui, Y., Belongie, S., and Hays, J. (2015). Learning deep representations for ground-to-aerial geolocalization. In *the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer.

Liu, H., Dai, J., Wang, R., Zheng, H., and Zheng, B. (2016a). Combining background subtraction and three-frame difference to detect moving object from underwater video. In *OCEANS 2016-Shanghai*, pages 1–5. IEEE.

Liu, H. and Hou, X. (2012). Moving detection research of background frame difference based on gaussian model. In *Computer Science & Service System (CSSS), 2012 International Conference on*, pages 258–261. IEEE.

Liu, M., Li, S., Shan, S., and Chen, X. (2015). AU-inspired deep networks for facial expression feature learning. *Neurocomputing*, 159:126–136.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. (2016b). SSD: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer.

Liu, Z.-G., Du, S.-Y., Yang, Y., and Ji, X.-H. (2014). A fast algorithm for color space conversion and rounding error analysis based on fixed-point digital signal processors. *Computers & Electrical Engineering*, 40(4):1405–1414.

López-Granados, F., Torres-Sánchez, J., Serrano-Pérez, A., de Castro, A. I., Mesas-Carrascosa, F.-J., and Peña, J.-M. (2016). Early season weed mapping in sunflower using UAV technology: variability of herbicide treatment maps against weed thresholds. *Precision Agriculture*, 17(2):183–199.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110.

Lukas, V., Novák, J., Neudert, L., Svobodova, I., Rodriguez-Moreno, F., Edrees, M., and Kren, J. (2016). The combination of UAV survey and landsat imagery for monitoring of crop vigor in precision agriculture. *ISPRS-International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, pages 953–957.

Maeda, H., Sekimoto, Y., Seto, T., Kashiyama, T., and Omata, H. (2018). Road damage detection using deep neural networks with images captured through a smartphone. *arXiv preprint arXiv:1801.09454*.

Maini, R. and Aggarwal, H. (2010). A comprehensive review of image enhancement techniques. *arXiv preprint arXiv:1003.4053*.

Montazer, G. A. and Giveki, D. (2015). An improved radial basis function neural network for object image retrieval. *Neurocomputing*, 168:221–233.

Moore, A., Allman, J., and Goodman, R. M. (1991). A real-time neural system for color constancy. *IEEE Transactions on Neural networks*, 2(2):237–247.

Okafor, E., Pawara, P., Karaaba, F., Surinta, O., Codreanu, V., Schomaker, L., and Wiering, M. (2016). Comparative study between deep learning and bag of visual words for wild-animal recognition. In *IEEE Symposium Series for Computational Intelligence (SSCI)*, pages 1–8. IEEE.

Okafor, E., Schomaker, L., and Wiering, M. A. (2018). An analysis of rotation matrix and colour constancy data augmentation in classifying images of animals. *Journal of Information and Telecommunication*, 2(4):465–491.

Okafor, E., Smit, R., Schomaker, L., and Wiering, M. (2017). Operational data augmentation in classifying single aerial images of animals. In *INnovations in Intelligent SysTems and Applications (INISTA), 2017 IEEE International Conference on*, pages 354–360. IEEE.

Parkhi, O. M., Vedaldi, A., and Zisserman, A. (2015). Deep face recognition. *Proceedings of the British Machine Vision Conference*, 1(3):6.

Passalis, N. and Tefas, A. (2016). Spectral clustering using optimized bag-of-features. In *Proceedings of the 9th Hellenic Conference on Artificial Intelligence*, pages 1–9. ACM.

Pawara, P., Okafor, E., Schomaker, L., and Wiering, M. (2017). Data augmentation for plant classification. In *International Conference on Advanced Concepts for Intelligent Vision Systems*, pages 615–626. Springer.

Petro, A. B., Sbert, C., and Morel, J.-M. (2014). Multiscale retinex. *Image Processing On Line*, pages 71–88.

Pietikäinen, M. (2010). Local binary patterns. *Scholarpedia*, 5(3):9775.

Pinto, N., Stone, Z., Zickler, T., and Cox, D. (2011). Scaling up biologically-inspired computer vision: A case study in unconstrained face recognition on Facebook. In *Computer Vision and Pattern Recognition Workshops (CVPRW), IEEE Computer Society Conference on*, pages 35–42.

Porto, S. M., Arcidiacono, C., Anguzza, U., and Cascone, G. (2013). A computer vision-based system for the automatic detection of lying behaviour of dairy cows in free-stall barns. *Biosystems Engineering*, 115(2):184–194.

Prathibha1, E., Manjunath, A., and Likitha, R. (2012). RGB to YCbCr color conversion using VHDL approach. *International Journal of Engineering and Development*, 1(3):15–22.

Provenzi, E., Marini, D., De Carli, L., and Rizzi, A. (2005). Mathematical definition and analysis of the retinex algorithm. *Journal of the Optical Society of America (JOSA) A*, 22(12):2613–2621.

Qing, C., Dickinson, P., Lawson, S., and Freeman, R. (2011). Automatic nesting seabird detection based on boosted HOG-LBP descriptors. In *Image Processing (ICIP), 2011 18th IEEE International Conference on*, pages 3577–3580. IEEE.

Rahman, Z.-u., Jobson, D. J., and Woodell, G. A. (1996). Multi-scale retinex for color image enhancement. In *Image Processing, 1996. Proceedings., International Conference on*, volume 3, pages 1003–1006. IEEE.

Rahman, Z.-u., Jobson, D. J., and Woodell, G. A. (2004). Retinex processing for automatic image enhancement. *Journal of Electronic Imaging*, 13(1):100–110.

Ramesh, B., Xiang, C., and Lee, T. H. (2015). Shape classification using invariant features and contextual information in the bag-of-words model. *Pattern Recognition*, 48(3):894–906.

Razavian, A., Azizpour, H., Sullivan, J., and Carlsson, S. (2014). CNN features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 806–813.

Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99.

Rogez, G. and Schmid, C. (2016). Mocap-guided data augmentation for 3d pose estimation in the wild. In *Advances in Neural Information Processing Systems*, pages 3108–3116.

Salton, G. et al. (1971). The smart retrieval system.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117.

Sengar, S. S. and Mukhopadhyay, S. (2017). Moving object detection based on frame difference and W4. *Signal, Image and Video Processing*, 11(7):1357–1364.

Sergyán, S. (2008). Color histogram features based image classification in content-based image retrieval systems. In *Applied Machine Intelligence and Informatics. SAMI 2008. 6th International Symposium on*, pages 221–224. IEEE.

Shin, H.-C., Roth, H. R., Gao, M., Lu, L., Xu, Z., Nogues, I., Yao, J., Mollura, D., and Summers, R. M. (2016). Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning. *IEEE transactions on medical imaging*, 35(5):1285–1298.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Sladojevic, S., Arsenovic, M., Anderla, A., Culibrk, D., and Stefanovic, D. (2016). Deep neural networks based recognition of plant diseases by leaf image classification. *Computational intelligence and neuroscience*, 2016.

Stutz, D. (2014). Understanding convolutional neural networks. In *Fakultät für Mathematik, Informatik und Naturwissenschaften Lehr- und Forschungsgebiet Informatik VIII Computer Vision*, pages 1–23.

Sun, K. B. and Super, B. J. (2005). Classification of contour shapes using class segment sets. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. Computer Society Conference on*, volume 2, pages 727–733. IEEE.

Surinta, O., Karaaba, M. F., Mishra, T. K., Schomaker, L. R. B., and Wiering, M. A. (2015). Recognizing handwritten characters with local descriptors and bags of visual words. In *Engineering Applications of Neural Networks*, pages 255–264. Springer.

Sutskever, I., Martens, J., Dahl, G. E., and Hinton, G. E. (2013). On the importance of initialization and momentum in deep learning. *International Conference on Machine Learning ICML (3)*, 28:1139–1147.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9.

Takahashi, K., Takahashi, S., Cui, Y., and Hashimoto, M. (2014). Remarks on computational facial expression recognition from HOG features using quaternion multi-layer neural network. In *Engineering Applications of Neural Networks*, pages 15–24. Springer.

Tang, S., Xu, Z.-X., Li, J.-T., and Zheng, Y.-T. (2015). An extremely efficient concept detection system via sparse ensemble learning. *Neurocomputing*, 169:124–133.

Tang, Y. (2013). Deep learning using linear support vector machines. In *Challenges in Representational learning, The ICML 2013 Workshop on.*

Van De Sande, K. E. A., Gevers, T., and Snoek, C. G. M. (2010). Evaluating color descriptors for object and scene recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(9):1582–1596.

Vedaldi, A. and Lenc, K. (2015). MatConvNet: Convolutional neural networks for Matlab. In *Proceedings of the 23rd Annual ACM Conference on Multimedia Conference*, pages 689–692.

Viola, P. and Jones, M. J. (2004). Robust real-time face detection. *International journal of computer vision*, 57(2):137–154.

Wang, C. and Huang, K. (2015). How to use bag-of-words model better for image classification. *Image and Vision Computing*, 38:65–74.

Wang, J., Yang, J., Yu, K., Lv, F., Huang, T., and Gong, Y. (2010). Locality-constrained linear coding for image classification. In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*, pages 3360–3367.

Welsh, T., Ashikhmin, M., and Mueller, K. (2002). Transferring color to greyscale images. In *ACM Transactions on Graphics (TOG)*, volume 21, pages 277–280.

Wilber, M. J., Scheirer, W. J., Leitner, P., Heflin, B., Zott, J., Reinke, D., Delaney, D. K., and Boult, T. E. (2013). Animal recognition in the Mojave desert: Vision tools for field biologists. In *Applications of Computer Vision (WACV), 2013 IEEE Workshop on*, pages 206–213. IEEE.

Ye, P., Kumar, J., Kang, L., and Doermann, D. (2012). Unsupervised feature learning framework for no-reference image quality assessment. In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*, pages 1098–1105.

Zhang, C. and Kovacs, J. M. (2012). The application of small unmanned aerial systems for precision agriculture: a review. *Precision agriculture*, 13(6):693–712.

Zhang, R., Isola, P., and Efros, A. A. (2016). Colorful image colorization. *arXiv preprint arXiv:1603.08511*.

Zhou, D. and Zhang, H. (2005). Modified GMM background modeling and optical flow for detection of moving objects. In *Systems, Man and Cybernetics, 2005 IEEE International Conference on*, volume 3, pages 2224–2229. IEEE.

SUMMARY

Detection and recognition of animals under different visual appearances is still a difficult problem within the computer vision community. This thesis analyzes the use of different deep learning techniques and conventional computer vision methods for performing animal recognition or detection with relatively small training datasets. The dissertation can be summarized into seven chapters; provides an introductory background, highlights the objectives of the thesis and areas for future works.

Chapter 1 provides a brief introductory background to understand animal recognition, detection, and image enhancement. Moreover, the objectives of the dissertation, the research questions, and significant contributions are explained.

Chapter 2 attempts to analyze the best image recognition method when there are not many images. For this, we propose the use of effectively reduced neural network systems compared to both existing network architectures (AlexNet or GoogleNet) and classical computer vision methods (variants of the bag of visual words (BOW or HOG-BOW)) combined with supervised learning algorithms for recognizing wild animals. In total, the 16 methods are used for performing the recognizing task on a relatively small wild-animal dataset. The results showed that deep learning methods either in the effectively reduced or existing forms significantly outperform the classical techniques. Our proposed methods present two merits when compared to the existing network architectures; it uses fewer network parameters during training of the network and lowers computational cost for some approaches; and secondly, the effectively reduced version of the AlexNet architecture when trained from scratch or pretrained schemes showed $\geq 26\%$ decrease in computational time. The results further demonstrated that the BOW method using color image intensities with the max-pooling strategy outperforms the HOG-BOW combined to either gray image intensities and the two spatial pooling strategies on the used dataset.

Chapter 3 analyzes the benefit of using our custom recognition system for recognizing and handling rotation variations in a novel unmanned aerial vehicle (UAV) image dataset without creating too many images. For this, we propose a rotation matrix algorithm as a novel method of data augmentation (DA). The new DA method is useful for enhancing the pixel information in an image. Additionally, the new DA method does not require an increase in the number of images during the training of the network compared to the conventional DA approaches. We trained several recognition systems: the reduced version of the GoogleNet with three inception modules while considering two classification loss functions (cross-entropy loss and hinge loss) using two training schemes (scratch and finetuned). Additionally, we investigated the classical computer vision techniques consisting of a local descriptor (HOG), a global descriptor (color histogram) and the combination of both on low or high-resolution image sizes. The results show that the DA methods combined with pretrained instances of the effectively reduced neural network system yield high classification scores.

Chapter 4 describes the development of recognition systems that are robust to illumination variations of different visual appearances. For this purpose, we developed a hybrid variant of the rotation-matrix data augmentation that combines rotation-matrix and color constancy as another method for DA. The proposed technique can be used to increase the number of training images especially when there exists an insufficient amount of images within a dataset. An additional merit of the proposed method is that it can enhance the illumination quality of a blurred image. We trained the recognition system (a customized version of the GoogleNet) with the cross-entropy loss function and compared the classification performances of the color-constancy-DA (with ORIG/ROT), ORIG alone, and ROT-DA alone on three datasets (UAV aerial images, Croatia Fish, and Bird-600). The results show that the finetuned CNN with an appropriate selection of the grid resolution and angular bounds for the rotation algorithm combined with color-constancy methods yields the highest classification accuracies on most of the used datasets. Moreover, the results show that using finetuned CNN models with the proposed data-augmentation (ROT-DA) technique on the Aerial UAV images leads to significantly better results than all other approaches. Finally, the results of our proposed approaches to data augmentation combined with the fine-tuned CNN surpass substantially previous research on the Bird-600 dataset.

Chapter 5 analyzes how important the use of color spaces is in deep learning. For this, we construct a color conversion algorithm that has the potential to transform a natural (RGB) or black and white (BW) image to four other color spaces. Then we employed our custom network to access the classification performances on several variants of the used animal datasets. We propose a color conversion algorithm, which presents the following merits: 1) It can transform binary masked (BW) images to images represented in different color spaces (RGB, YCbCr, HSV, Lab), which may aid to boost the CNN classification performance, and 2) It is an efficient algorithm and easy to implement or use. The results show that training a CNN on artificially colorized images or the original BW images, yields almost similar performance levels on the Animal-Shape dataset and this indicates that the use of different spaces could be an alternative. Still, the results of the CNN architecture on the MPEG-7 dataset show that our system obtains the second best-reported result on this dataset. Lastly, the results of the CNN applied to the Wild-Anim dataset show that the use of the natural RGB color space is essential, as it obtains significantly better results than using other color spaces.

Chapter 6 analyzes detection algorithms for detecting and recognizing individual instances of badgers. The problem here is that the animals (badgers) under study exhibit similarities in color appearance, and therefore using classical approaches may not be very suitable for discriminating between the several individual instances of this specific animal. One possible alternative to tackle this problem is to compare two neural network based detectors: SSD and Faster R-CNN. SSD is combined with the Inception-V2 or MobileNet as a backbone, and the Faster R-CNN detector is combined with either Inception-V2 or Residual networks with 50 layers (ResNet-50) as feature extractors. Furthermore, we compare the use of two output activation functions: the softmax and sigmoid function. For the experiments, we use several videos recorded with a low-resolution camera. The results show that most of the trained SSD detectors significantly outperform the different variants of the Faster R-CNN detector. All the Faster R-CNN methods are computationally much faster than the SSD techniques for training the system, although for testing SSD is a bit faster.

Chapter 7 provides the lessons learned from each of the stated objectives of this thesis and highlights possible areas for future work. An area of further research is to employ a segmentation based algorithm for segmenting regions of interest from a localized region, which can be very useful to improve a recognition and detection system.

SAMENVATTING

Detectie en herkenning van dieren in verschillende visuele verschijningen is nog altijd een moeilijk probleem in de computer vision-gemeenschap. Deze scriptie onderzoekt de toepassing van verschillende deep learning-technieken en conventionele computer vision-methodes voor herkenning en detectie van dieren met relatief kleine datasets voor training. De dissertatie wordt samengevat in zeven hoofdstukken; het geeft een inleidende achtergrond, benoemt het doel van de scriptie, en geeft suggesties voor verder onderzoek.

Hoofdstuk 1 verschaft een korte introducerende achtergrond om het herkennen en detecteren van dieren, en verbeteren van afbeeldingen te begrijpen. Verder worden de doelen van de dissertatie, de onderzoeksvraag, en significante bijdragen uitgelegd.

Hoofdstuk 2 biedt een analyse van de beste methode voor beeldherkenning in situaties met weinig afbeeldingen. Hiervoor stellen we voor om gebruik te maken van effectief gereduceerde neurale netwerken, vergeleken met zowel bestaande netwerkarchitecturen (AlexNet en GoogleNet) als klassieke computer vision-methodes (varianten van de bag of visual words (BOW of HOG-BOW)), gecombineerd met supervised learning-algoritmes voor het herkennen van wilde dieren. In totaal worden 16 methodes gebruikt om de herkenningstaak uit te voeren op een relatief kleine dataset van wilde dieren. Uit de resultaten blijkt dat de deep learning-methodes in zowel de effectief gereduceerde als de bestaande vorm significant beter presteren dan de klassieke technieken. Onze voorgestelde methode heeft twee voordelen ten opzichte van de bestaande netwerkarchitecturen: er zijn minder netwerkparameters nodig tijdens de training van het netwerk en in sommige situaties is minder rekenkracht nodig; en ten tweede wordt de rekentijd verlaagd met $\geq 26\%$ bij de effectief gereduceerde versie van de AlexNet-architectuur, indien getraind from scratch of met een pretrained systeem. Ook blijkt uit de resultaten dat de BOW-methode gebruikmak-

end van de intensiteiten van kleurenafbeeldingen met de max-pooling-strategie, beter presteert dan de HOG-BOW-methode met intensiteiten van zwart-witafbeeldingen met de twee ruimtelijke pooling-strategieën op de gebruikte dataset.

Hoofdstuk 3 analyseert de voordelen van een op maat gemaakt herkenningssysteem voor herkenning en verwerking van geroteerde varianten van een nieuwe dataset met afbeeldingen van een onbemand luchtvaartuig (unmanned aerial vehicle; UAV), zonder te veel afbeeldingen te creëren. Hiervoor stellen we voor om een rotatiematrixalgoritme als nieuwe methode voor data augmentation (DA) te gebruiken. Deze nieuwe DA-methode is nuttig voor het verbeteren van de pixelinformatie in een afbeelding. Bovendien zorgt de nieuwe DA-methode niet voor een toename van het aantal afbeeldingen voor training in het netwerk vergeleken met conventionele DA-aanpakken. We hebben verschillende herkenningssystemen getraind: de gereduceerde versie van GoogleNet met drie inception-modules, waarbij we twee loss-functies voor de classificatie (cross-entropy loss en hinge loss) en twee trainingsmethodes (scratch en finetuned) vergelijken. Bovendien hebben we de klassieke computer vision-technieken onderzocht, bestaande uit een local descriptor (HOG), een global descriptor (kleurhistogram), en de combinatie van beide op afbeeldingen van zowel lage als hoge resolutie. Uit de resultaten blijkt dat de DA-methodes gecombineerd met pretrained systemen van het effectief gereduceerde neurale netwerk hoge classificatiescores behaalt.

Hoofdstuk 4 beschrijft de ontwikkeling van herkenningssystemen die robuust zijn tegen varianten in belichting of verschillende visuele verschijningen. Hiervoor hebben we een hybride variant van het rotatiematrix-DA-algoritme ontwikkeld die de rotatiematrix en kleurvastheid combineert als andere DA-methode. Deze voorgestelde techniek kan gebruikt worden om het aantal trainingsafbeeldingen te vergroten wanneer een dataset onvoldoende afbeeldingen bevat. Een extra voordeel van de voorgestelde methode is dat deze de belichtingskwaliteit van onscherpe afbeeldingen kan verbeteren. We hebben een herkenningssysteem getraind (een aangepaste versie van GoogleNet) met de cross-entropy loss-functie en de classificatieprestaties vergeleken van de kleurconstantie-DA (met ORIG/ROT), enkel ORIG, en enkel ROT-DA op drie datasets (UAV-luchtfotos, Croatia

Fish, en Bird-600). Uit de resultaten blijkt dat de finetuned CNN met een geschikte instelling van de rasterresolutie en hoekgrenzen voor het rotatiealgoritme, gecombineerd met kleurconstantiemethodes de hoogste classificatienauwkeurigheid behaalt bij de meeste van de gebruikte datasets. Bovendien blijkt uit de resultaten dat het gebruik van finetuned CNN-modellen met de voorgestelde DA-techniek (ROT-DA) voor de UAV-luchtfotȯs tot significant betere resultaten leidt dan alle andere aanpakken. Tenslotte blijkt dat de resultaten van onze voorgestelde DA-aanpak gecombineerd met de finetuned CNN, de resultaten van eerder onderzoek op de Bird-600-dataset substantieel overtreffen.

Hoofdstuk 5 analyseert het belang van kleurruimtes in deep learning. Hiervoor bouwen we een kleurconversiealgoritme waarmee natuurlijke (RGB) en zwart-wit (BW) afbeeldingen naar vier andere kleurruimtes getransformeerd kunnen worden. Daarna hebben we ons op maat gemaakte netwerk ingezet om de classificatieprestaties op verschillende varianten van de gebruikte dierendatasets te meten. We stellen een kleurconversiealgoritme met de volgende eigenschappen voor: 1) het kan binary masked (BW) afbeeldingen transformeren naar afbeeldingen in verschillende kleurruimtes (RGB, YCbCr, HSV, en Lab), wat de classificatieprestatie van het CNN kan verbeteren, en 2) het is een efficient algoritme dat eenvoudig te implementeren of gebruiken is. Uit de resultaten blijkt dat het trainen van een CNN op kunstmatig gekleurde afbeeldingen of de originele BW-afbeeldingen bijna gelijke prestatieniveaus op de Animal-Shape-dataset haalt, wat aanduidt dat het gebruik van verschillende kleurruimtes een alternatief kan zijn. Uit de resultaten van de CNN-architectuur op de MPEG-7-dataset blijkt dat ons systeem het op een na beste resultaat behaalt dat is gerapporteerd voor deze dataset. Tenslotte blijkt uit de resultaten van de CNN toegepast op de Wild-Anim-dataset, dat het gebruik van de natuurlijke RGB-kleurruimte essentieel is omdat deze significant betere resultaten behaalt dan andere kleurruimtes.

Hoofdstuk 6 analyseert detectiealgoritmes voor detectie en herkenning van individuele exemplaren van dassen. Het probleem is dat de dieren (dassen) in het onderzoek vergelijkbare kleurverschijningen hebben, en de klassieke aanpakken hierom mogelijk niet geschikt zijn voor het onderscheiden van verschillende individuele exemplaren van dit specifieke dier. Als mogelijk alternatief om dit probleem aan te pakken vergelijken we twee detectors gebaseerd op neurale netwerken: SSD en Faster R-CNN. SSD wordt gecombineerd met Inception-V2 of MobileNet als backbone, en de Faster R-CNN-detector wordt gecombineerd met Inception-V2 of een residual netwerk met 50 lagen (ResNet-50) als feature extractor. Bovendien vergelijken we het gebruik van twee output activation-functies: de softmax- en sigmoid-functie. Voor de experimenten gebruiken we verschillende video′s die zijn opgenomen met een lageresolutiecamera. Uit de resultaten blijkt dat de meeste van de getrainde SSD-detectors significant beter presteren dan de verschillende varianten van de Faster R-CNN-detectors. Alle Faster R-CNN-methodes zijn in rekentijd veel sneller dan de SSD-technieken bij het trainen van het systeem, terwijl SSD voor het testen een beetje sneller is.

Hoofdstuk 7 behandelt de geleerde lessen voor elk van de genoemde doelen van deze scriptie, en suggesties voor verder onderzoek. Een gebied voor verder onderzoek is het gebruik van segmentatie-gebaseerde algoritmes om regions of interest te segmenteren van lokale gebieden, wat erg nuttig kan zijn om herkennings- en detectiesystemen te verbeteren.

ACKNOWLEDGEMENTS

The success of my PhD wouldn't have been possible without the collaborative efforts and support of many people.

Immeasurable appreciations and thanks go to my supervisors (mentors) Dr. Marco Wiering and Prof. Lambert Schomaker, whose constructive critics and guidance has developed me to become an independent researcher. Their tremendous academic insights have aided in the actualization of this thesis. I greatly thank my promoter Prof. Lambert Schomaker for all his support and a one-year employment contract during the MANTIS project. I am grateful to Dr. Marco Wiering who has expert knowledge in machine learning and reinforcement learning. He was my first contact here in the Netherlands and later became my daily supervisor. He often motivated me and shared many interesting ideas.

I want to thank all the assessment committee members for their insightful remarks on my dissertation. Many thanks go to Jonathan Hogervorst who assisted me in translating the summary section of my thesis from English to Dutch. Additionally, I want to thank Remco Wouts and the Center for Information Technology, the University of Groningen for their support on this research. My profound appreciations go to my lovely wife; Joy, parents (Sir Wilfred and Joy Okafor (late)), siblings (Dr. Ekene, Uche and Angela), my uncle (Engr. Ben Okafor), and in-laws for their prayers and emotional support during my doctoral research.

Special thanks to the Ahmadu Bello University (ABU), Zaria for granting me a study leave fellowship to pursue a doctorate programme. My immense gratitude goes to Prof. Muhammed Bashir Muazu, Dr. Yusuf Jibril, and all my academic colleagues in the Department of Computer, Electrical and Communication Engineering, ABU, for their active collaboration.

I thank my paranimfen Pry and Oscar who helped in the PhD ceremonial preparation. Additionally, I enjoyed discussion with my office mates; Faik, Klaas, Sha, Mahya, and Jean-Paul, and my co-authors; Gerard, Olarik, Valeriu, Sukalpa, Rik, Martijn, Jonathan Hogervorst, Floris, Arvid, and

Jonathan Maas, for the lovely time and ideas shared during my research and other collaborative projects.

I want to thank the Secretaries of the Bernoulli Office; Elina Sietsema and Sarah van Wouwe, who were very kind and helpful, I do sincerely appreciate all your support. I appreciate my colleagues from the Artificial Intelligence/Cognitive Modeling group; Stefan, Primoz, Ben, Stipe, David, Maruf, Anja, Hermine, and Anna for the nice chatting during lunch. I do appreciate my friends outside the department; Musty, Chima, Ikechi, and Edmond for the friendly and quality time spent here in Groningen.

To all my academic colleagues and members of staff in the Bernoulli Institute for Mathematics, Computer Science and Artificial Intelligence at the University of Groningen not listed here, I am delighted to have worked with you all.

AUTHOR PUBLICATIONS

Thesis Publications

- **Okafor. E.**, Berendsen. G., Schomaker. L.R.B., and Wiering. M. A (2018). Detection and Recognition of Badgers using Deep Learning, *In 27th International Conference on Artificial Neural Networks (ICANN), Lecture Notes in Computer Science book series; vol. 11141, pages 554-563. Springer, Cham.*

- **Okafor. E.**, Schomaker. L.R.B., and Wiering. M. A (2018). An Analysis of Rotation-Matrix and Color Constancy Data Augmentation in Classifying Images of Animals, *Journal of Information and Telecommunication, ISSN 2475-1839, Vol 2: 4, pages 465-491.*

- **Okafor. E.**, Smit. R., Schomaker. L.R.B., and Wiering. M. A (2017). Operational Data Augmentation in Classifying Single Aerial Images of Animals, *In INnovations in Intelligent SysTems and Applications (INISTA), 2017 IEEE International Conference on, pages 354-360. IEEE.*

- **Okafor. E.**, Pawara. P., Karaaba. F, Surinta. O., Codreanu. V., Schomaker. L.R.B., and Wiering. M. A (2016). Comparative Study Between Deep Learning and Bag of Visual Words for Wild-Animal Recognition, *In IEEE Symposium Series on Computational Intelligence (SSCI) on, pages 1-8. IEEE.*

Other Publications

- Beers. v. F., Lindstrom. A., **Okafor. E.** and Wiering. M. A (2019). Deep Neural Networks with Intersection over Union Loss for Binary Image Segmentation, *In 8th International Conference on Pattern Recognition Applications and Methods (ICPRAM).*

- Sillitti A., Anakabe J. F., Basurko J., Dam P., Ferreira H., Ferreiro S., Gijsbers J., He S., Hegedus C., Holenderski M., Hooghoudt J., Lecuona I., Leturiondo U., Marcelis Q., Moldovan I., **Okafor E.,** Rebelo de Sa C, Romero R, Sarr B, Schomaker L, Shekar A. K., Soares C., Sprong H., Theodorsen S., Tourwe T., Urchegui G., Webers G., Yang Y., Zubaliy A., Zugasti E., Zurutuza U. Providing Proactiveness: Data Analysis Techniques Portfolios (2019). *The MANTIS Book Cyber Physical System Based Proactive Collaborative Maintenance;on, pages 145-238. ISBN 978-87-93609-84-6; Rivers Publishers.*

- Chanda. S., **Okafor. E.**, Hamel S., Stutzmann D., and Schomaker. L.R.B (2018). Deep Learning for Classification and as Tapped-Feature Generator in Medieval Word-Image Recognition, *In 2018 13th IAPR International Workshop on Document Analysis Systems (DAS) on, pages. 217-222. IEEE.*

- Hogervorst. J., **Okafor. E.**, and Wiering. M. A (2017). Deep Colorization for Facial Gender Recognition. *In Preproceedings of the 29th Benelux Conference on Artificial Intelligence (BNAIC'2017) on, pages 317-325.*

- Pawara. P., **Okafor. E.**, Schomaker. L.R.B., and Wiering. M. A (2017). Data Augmentation for Plant Classification. *In International Conference on Advanced Concepts for Intelligent Vision Systems on, pages 615-626, Springer.*

- Pawara. P, **Okafor. E.**, Surinta. O., Schomaker. L.R.B., and Wiering. M.A (2017). Comparing Local Descriptors and Bags of Visual Words to Deep Convolutional Neural Networks for Plant Recognition, *In 6th International Conference on Pattern Recognition Applications and Methods (ICPRAM) on, pages 479-486.*

- Wagenaar. M., **Okafor. E.**, Frencken. W., and Wiering. M.A (2017). Using Deep Convolutional Neural Networks to Predict Goal-scoring Opportunities in Soccer, *In 6th International Conference on Pattern Recognition Applications and Methods (ICPRAM) on, pages 448-455.*

- Maas. J. L., **Okafor. E.**, and Wiering. M. A(2016). The Dual Codebook: Combining Local Feature Descriptors for Bags of Visual

Words in Image Classification, *In Preproceedings of the 28th Annual Benelux Conference on Artificial Intelligence (BNAIC 2016) on, pages 64-71.*