Comparative Study Between Deep Learning and Bag of Visual Words for Wild-Animal Recognition

Emmanuel Okafor*, Pornntiwa Pawara*, Faik Karaaba*, Olarik Surinta‡,
Valeriu Codreanu†, Lambert Schomaker* and Marco Wiering* (IEEE Member)
*Institute of Artificial Intelligence and Cognitive Engineering,
University of Groningen, The Netherlands
†Surf Sara BV, Science Park 140, Amsterdam, The Netherlands
‡Multi-Agent Intelligent Simulation Laboratory (MISL), Mahasarakham University, Thailand

***Abstract*—Most research in image classification has focused on applications such as face, object, scene and character recognition. This paper examines a comparative study between deep convolutional neural networks (CNNs) and bag of visual words (BOW) variants for recognizing animals. We developed two variants of the bag of visual words (BOW and HOG-BOW) and examine the use of gray and color information as well as different spatial pooling approaches. We combined the final feature vectors extracted from these BOW variants with a regularized L2 support vector machine (L2-SVM) to distinguish between classes within our datasets. We modified existing deep CNN architectures: AlexNet and GoogleNet, by reducing the number of neurons in each layer of the fully connected layers and last inception layer for both scratch and pre-trained versions. Finally, we compared the existing CNN methods, our modified CNN architectures and the proposed BOW variants on our novel wild-animal dataset (Wild-Anim). The results show that the CNN methods significantly outperform the BOW techniques.**

I. Introduction

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 paper, 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 [1]. In this paper, we will focus on 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 [2]–[4]. Also, the authors in [5] 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 (BOW) has been applied to different fields. BOW involves the extraction of features [6], [7] and construction of a codebook using an unsupervised learning algorithm such as K-means clustering [8], spectral clustering [9], local constrained linear coding for pooling clusters [10], and the use of the fast minimum spanning tree [11]. Finally, the extraction of feature vectors by the BOW approach can be achieved using a soft assignment scheme [12] or sparse ensemble learning methods [13]. Some recent works have used BOW as an input to some hierarchical structures such as weakly supervised deep metric learning [14] and robust structured subspace learning [15]. 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 [16] and face recognition [17]. In [18], 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 [19], [20], object recognition [21], handwritten character recognition [22], [23] and medical image recognition [24]. The use of deep learning to learn from large datasets has led to the evolution of deep architectures like AlexNet [25], GoogleNet [26] and Residual Networks (ResNets) [27].

The BOW method [6] has been a popular and widely used method in the computer vision community. According to [28], 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 [29] 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 [30] demonstrated that CNN augmentation with a support vector machine (SVM) outperforms BOW and other local feature descriptors.

Contributions: In this paper, we evaluate the performance of 16 different techniques on a novel wild-animal dataset. To actualize this aim, the use of existing deep CNN architectures (GoogleNet and AlexNet), our modified versions of the deep CNN (Reduced Fine-tuned and Scratch versions of GoogleNet and AlexNet) and variants of BOW techniques are applied to our novel Wild-Anim dataset. The results show that our 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 our 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 our 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 [16] and facial recognition [17].

Paper Outline: This paper is organized in the following way. Section II briefly explains the basic deep learning processes. Section III describes the different learning techniques we use in our wild animal recognition system. Section IV 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 V. The conclusion is presented in Section VI.

II. 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) [25] 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 [25], [31]. 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 [32] 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 [25] 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 [25] regularization scheme is applied to the output of the spatial pooling layer.

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 [33] 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 [31]:

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

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 our experiments.

III. Learning Methods

We study 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 our 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 our modifications, resulting in 8 different deep learning architectures.

A. *AlexNet Architecture*

The AlexNet model, initially proposed in [25], 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. In this paper, we explore the use of both original and reduced versions of both scratch and fine-tuned AlexNet models on our Wild-Anim dataset. Our experimental procedure is similar to that of [25], that applied the stochastic gradient descent update rule with momentum, which is expressed as:

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

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

where W_i are the weights of the CNN, u_i is the weight change, L is the softmax loss function, 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 our dataset in order to obtain models that can be used to evaluate on our 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 (30 epochs) for 1000 snapshots. 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 our 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 Fig. 1 illustrates the modified version of the AlexNet architecture. The choice of

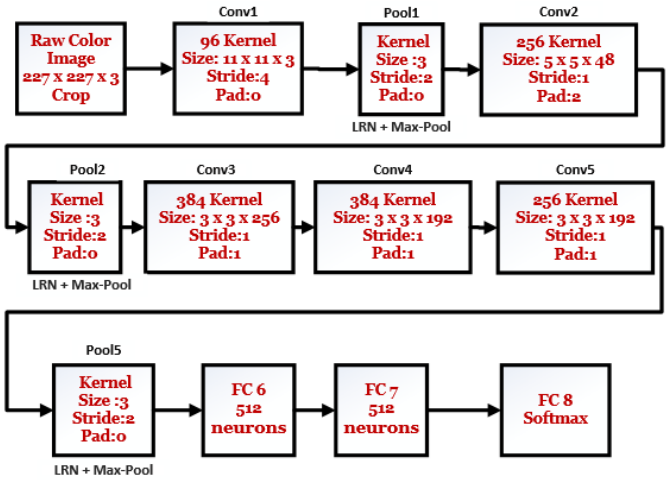

Fig. 1. Block Diagram of Modified AlexNet Architecture with Reduction in Neurons in the Fully Connected Layers.

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 our dataset.

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)) [25]. 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 our 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 epochs) with a step size of 10000 using a fixed learning rate of 0.001. We note that all our experiments were carried out using the Caffe platform on a Ge-Force GTX 960 GPU model.

B. GoogleNet Architecture

The GoogleNet deep learning architecture, proposed in [26], is one of the most powerful deep CNN models. This method is inspired from the incorporation of a new module known as inception which allows the concatenation of filters of different dimensions and sizes into a single new filter [24]. GoogleNet consists of two outer convolutional layers, two outer pooling layers, three sets of top-1 and top-5 loss-functions for three classifiers with a regularization dropout of 0.7, 0.7 and 0.4 respectively for the three classifiers and nine inception layers. In each inception layer, there exist six convolution layers and one pooling layer. Our experimental procedure is similar to that of [26] that employs gradient descent and the momentum update rule. We will briefly explain the four GoogleNet architectures that we will use in our experiments.

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 epochs) for 1000 snapshots. The number of output neurons in 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 output 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 neurons in each layer in the last inception layer: 24, 24, 24, 16 and 16 respectively, except for the first layer which has 384 neurons. The block diagram in Fig. 2 illustrates 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 epochs) with a step size of 10000. We used a fixed learning rate of 1×10^{-3}.

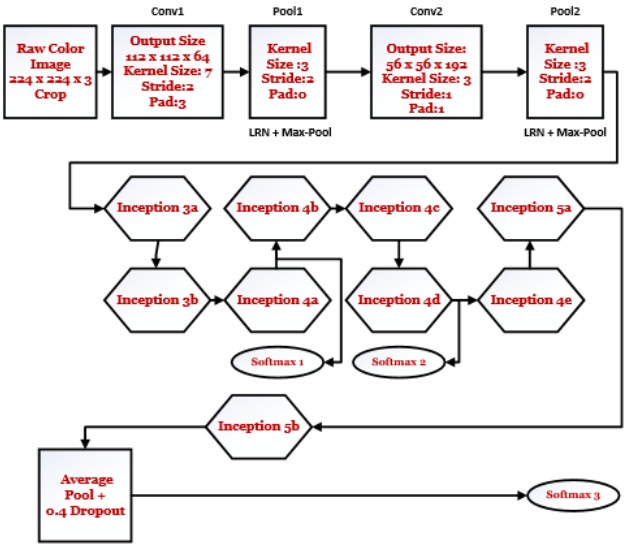

Fig. 2. Block Diagram of RS-GoogleNet Architecture Showing also the 3 Softmax Classifiers.

C. 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 [8] using K-means clustering. The steps involved in setting up BOW consist of three processes which we will explain now.

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 our 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 [8], where k is the number of centroids. In our preliminary experiments, we used randomly selected patches to compute the codebook. Our final choice was the use of 100,000 patches, because this extracts most information from our 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 [28] 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 [28]. 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 [34]. 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 [35], [36], 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) [37]. 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 [34]. In our 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 our 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 our 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 our 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 [38] is computed using:

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

We use the regularized L2-SVM [39] 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.

IV. Animal Dataset and Pre-Processing

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

A. *Wild-Anim Dataset*

We collected our novel dataset by downloading images of animals from Flicker. Our 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. Our dataset is processed by automatic labelling and then was normalized to 250 $\times$ 250 pixels introducing slightly anamorphic distortions. All images in this dataset are in RGB color space. A sample of the images in our dataset is shown in Fig. 3. After collecting our dataset, we noticed that ImageNet also contains the same

Fig. 3. Samples of the Images in the Wild-Anim Dataset, from left column to right column: lion, wolf, bear, elephant and leopard.

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 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.

V. Experimental 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.

A. *Evaluation of 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 I 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 I
Performances of the 16 Different Techniques On Our 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

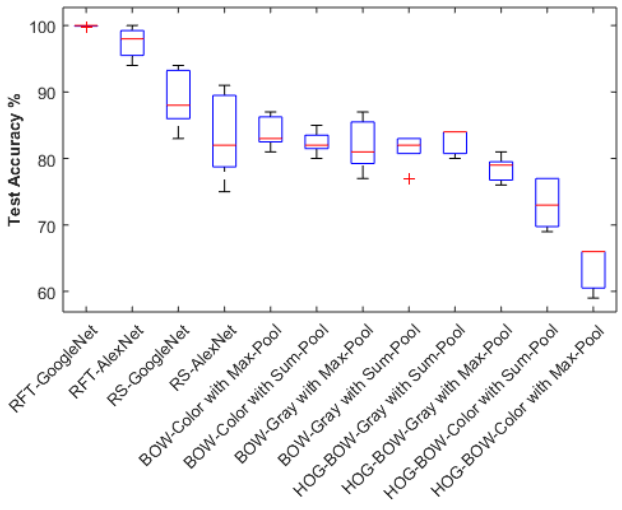

Fig. 4. Performance Evaluation of Our Modified Versions of Deep CNN Architectures and the BOW Variants on 5 Test Sets.

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 Fig 4. 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 I, 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 I 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 which outperforms OS-AlexNet by 1.6%, RS-AlexNet by 0.6% and HOG-BOW-gray with sum-pooling. This may be because there is 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 CNN results, 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 $10\times$ 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 II. The results are fairly consistent compared to the earlier results reported in Table I. This implies that the reduced Deep CNN architectures have an outstanding generalization.

TABLE II
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

VI. CONCLUSION

In this paper, we have compared many different image recognition techniques on a novel dataset consisting of wild animals. From the results, we conclude 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 to a big surprise. When we compare the performances of GoogleNet and AlexNet when trained from scratch, then GoogleNet performs much better. It is remarkable that the recognition accuracy is still very high even for the used small dataset.

We have also been able to demonstrate 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.

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

REFERENCES

[1] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, 2015.

[2] K. E. A. Van De Sande, T. Gevers, and C. G. M. Snoek, "Evaluating color descriptors for object and scene recognition," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 32, no. 9, pp. 1582–1596, 2010.

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

[4] S. Sergyán, "Color histogram features based image classification in content-based image retrieval systems," in *Applied Machine Intelligence and Informatics. SAMI 2008. 6th International Symposium on*. IEEE, 2008, pp. 221–224.

[5] A. Khosla, J. Xiao, A. Torralba, and A. Oliva, "Memorability of image regions," in *Advances in Neural Information Processing Systems*, 2012, pp. 305–313.

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

[7] C. Wang and K. Huang, "How to use bag-of-words model better for image classification," *Image and Vision Computing*, vol. 38, pp. 65–74, 2015.

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

[9] N. Passalis and A. Tefas, "Spectral clustering using optimized bag-of-features," in *Proceedings of the 9th Hellenic Conference on Artificial Intelligence*. ACM, 2016, pp. 1–9.

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

[11] R. Jothi, S. K. Mohanty, and A. Ojha, "Fast minimum spanning tree based clustering algorithms on local neighborhood graph," in *Graph-Based Representations in Pattern Recognition*. Springer, 2015, pp. 292–301.

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

[13] S. Tang, Z.-X. Xu, J.-T. Li, and Y.-T. Zheng, "An extremely efficient concept detection system via sparse ensemble learning," *Neurocomputing*, vol. 169, pp. 124–133, 2015.

[14] Z. Li and J. Tang, "Weakly supervised deep metric learning for community-contributed image retrieval," *IEEE Transactions on Multimedia*, vol. 17, no. 11, pp. 1989–1999, 2015.

[15] Z. Li, J. Liu, J. Tang, and H. Lu, "Robust structured subspace learning for data representation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 10, pp. 2085–2098, 2015.

[16] O. Surinta, M. F. Karaaba, T. K. Mishra, L. R. B. Schomaker, and M. A. Wiering, "Recognizing handwritten characters with local descriptors and bags of visual words," in *Engineering Applications of Neural Networks*. Springer, 2015, pp. 255–264.

[17] M. F. Karaaba, O. Surinta, L. R. B. Schomaker, and M. A. Wiering, "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*, 2016, pp. 582–589.

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

[19] N. Pinto, Z. Stone, T. Zickler, and D. Cox, "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*, 2011, pp. 35–42.

[20] O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Deep face recognition," *Proceedings of the British Machine Vision Conference*, vol. 1, no. 3, p. 6, 2015.

[21] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds., 2012, pp. 1097–1105.

[22] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient based learning applied to document recognition," in *Proceedings of the IEEE*, vol. 86(11), 1998, pp. 2278–2324.

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

[24] H.-C. Shin, H. R. Roth, M. Gao, L. Lu, Z. Xu, I. Nogues, J. Yao, D. Mollura, and R. M. Summers, "Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning," *IEEE transactions on medical imaging*, vol. 35, no. 5, pp. 1285–1298, 2016.

[25] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[26] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1–9.

[27] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," *arXiv preprint arXiv:1603.05027*, 2016.

[28] A. Coates, A. Y. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," in *International conference on artificial intelligence and statistics*, 2011, pp. 215–223.

[29] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[30] A. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "CNN features off-the-shelf: an astounding baseline for recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2014, pp. 806–813.

[31] A. Vedaldi and K. Lenc, "MatConvNet: Convolutional neural networks for Matlab," in *Proceedings of the 23rd Annual ACM Conference on Multimedia Conference*, 2015, pp. 689–692.

[32] D. Stutz, "Understanding convolutional neural networks," in *Fakultät für Mathematik, Informatik und Naturwissenschaften Lehr- und Forschungsgebiet Informatik VIII Computer Vision*, 2014, pp. 1–23.

[33] I. G. Y. Bengio and A. Courville, "Deep learning," 2016, book in preparation for MIT Press.

[34] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition (CVPR). IEEE Computer Society Conference on*, vol. 1, 2005, pp. 886–893.

[35] O. L. Junior, D. Delgado, V. Gonçalves, and U. Nunes, "Trainable classifier-fusion schemes: an application to pedestrian detection," in *Intelligent Transportation Systems*, vol. 2, 2009.

[36] K. Takahashi, S. Takahashi, Y. Cui, and M. Hashimoto, "Remarks on computational facial expression recognition from HOG features using quaternion multi-layer neural network," in *Engineering Applications of Neural Networks*. Springer, 2014, pp. 15–24.

[37] J. Arróspide, L. Salgado, and M. Camplani, "Image-based on-road vehicle detection using cost-effective histograms of oriented gradients," *Journal of Visual Communication and Image Representation*, vol. 24, no. 7, pp. 1182–1190, 2013.

[38] Y. Tang, "Deep learning using linear support vector machines," in *Challenges in Representational learning, The ICML 2013 Workshop on*, 2013.

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