EasyChair Preprint

№ 6666

Animal Detection for Road Safety Using Deep Learning

S Sanjay, Sudhir Sidhaarthan Balamurugan and
Sai Sudha Panigrahi

October 21, 2021

Animal Detection for Road safety using Deep Learning

Sanjay S[1]
RMK Engineering College
Chennai, India

Sudhir Sidhaarthan B[2]
Lovly Professional University
Punjab, India

Sai Sudha Panigrahi[3]
Amrita School of Engineering,
Coimbatore, India

***Abstract*— The recognition of big animals on the images with road scenes has received little attention in modern research. There are very few specialized data sets for this task. Popular open data sets contain many images of big animals, but the most part of them is not correspond to road scenes that is necessary for on-board vision systems of unmanned vehicles. The paper describes the preparation of such a specialized data set based on Google Open Images and COCO datasets. The resulting data set contains about 20000 images of big animals of 10 classes: "Bear", "Fox", "Dog", "Horse", "Goat", "Sheep", "Cow", "Zebra", "Elephant", "Giraffe". Deep learning approaches to detect these objects are researched in the paper. Authors trained and tested modern neural network architectures YOLOv3, RetinaNet R-50-FPN, Faster R-CNN R-50-FPN, Cascade RCNN R-50-FPN. To compare the approaches the mean average precision (mAP) was determined at IoU≥50%, also their speed was calculated for input tensor sizes 640x384x3. The highest quality metrics are demonstrated by architecture YOLOv3 as for ten classes (0.78 mAP) and one joint class (0.92 mAP) detection with speed more 35 fps on NVidia Tesla V-100 32GB video card. At the same hardware, the RetinaNet R-50-FPN architecture provided recognition speed of more than 44 fps and a 13% lower mAP. The software implementation was done using the Keras and PyTorch deep learning libraries and NVidia CUDA technology. The proposed data set and neural network approach to recognizing big animals on images have shown their effectiveness and can be used in the on-board vision systems of driverless cars or in driver assistant systems.**

Keywords*— *image recognition, detection, big animals, road scene, data set, deep learning, neural network, software.

I. INTRODUCTION

Reliable detection of big animals on images is a serious challenge for the computer vision systems of unmanned cars. This is especially important because of the relatively high number road accidents with wild animals[1].

At the early stage, approaches to solving this problem were used detectors based on hand-crafted features: Haarfeatures, HOG (Histogram of oriented gradients), LBP (Local binary patterns) [2, 3]. However, such approaches were not reliable enough.

Modern research in the field of big animals detection on images is associated, mainly, with the usage of deep convolutional neural networks. Moreover, the recognition of animals is investigated as a solution to the problems of classification [4], detection [5] and segmentation [6] of objects. Some works are devoted to the detection of animals on images obtained from unmanned aerial vehicles, for example, paper [7].

The appearance of animals on the road is a relatively rare event, at the same time, sufficiently large and varied data sets are needed to train neural network systems for their detection.

Table I shows the most popular modern open data sets containing images for the detection of big animals. There are also closed data sets created on the basis of images and videos from the Internet, for example, LADSet [3], but there is little information about their contents.

The IWildCam [1], Animal Image [2], The Oxford-IIITPet [3], and STL-10 [4] datasets have disadvantage that they contain a small number of labeled images in the training set and a limited number of animal classes. The largest ImageNet image database [5] currently contains many labeled images of a huge number of types and subtypes of big animals, but the vast majority of them do not apply to the road scene.

TABLE I. OPEN DATA SETS FOR BIG ANIMAL'S DETECTION PROBLEM

Data sets	**Total amount of images in the data set**
IWildCam [1]	~200k
Animal Image [2]	3k
The Oxford-IIIT-Pet [3]	7.5k
STL-10 [4]	100k
ImageNet [5]	14kk
COCO [6]	330k
Google's Open Images [7]	1.9kk

The COCO [6] and Google's Open Images [7] data sets are more promising for use in the research area, and they contain not only bounding boxes, but also polygons of object segments. In the present article, in section III, we consider the formation on their basis of a data set for the detection of big animals on the road scene.

In addition, special attention is paid to the use of modern object detectors based on deep convolutional neural networks

and the results of experiments using the created data set are analyzed.

II. Problem Definition

This article solves the problem of detecting and classifying animals on the image with road scene. We need to investigate methods based on deep neural networks for detection big animals of 10 widespread classes: “Bear”, “Fox”, “Dog”, “Horse”, “Goat”, “Sheep”, “Cow”, “Zebra”, “Elephant”, “Giraffe”. Also task includes the need to study the detection of an one joint class. To train and test various neural network architectures appropriate data set should be generated. Then we need to determine the best architecture for this task with AP (average precision) [15] quality metric per class and overall mAP (mean average precision) [16]. Another important indicator is the inference time for one image (without taking into account the loading time of the image into memory and its preparation for supplying the network input).

III. Data set Preparation

To obtain specific results, we created our own data set based on COCO [13] and Google’s Open Images V5 [14]. The following classes of large animals were selected from COCO data set: “Dog”, “Horse”, “Sheep”, “Cow”, “Bear”, “Elephant”, “Zebra”, “Giraffe”. Although there are almost no representatives of the last 3 classes in the area under consideration, they were added to improve the quality of the future detector by their recognition on the road scene. Open Images V5 contains previous and additional two classes of large animals: deer, “fox” and “goat”. Annotations to images are stored in COCO format, i.e. are contained in the .json file. Let's consider in more detail which fields are included in it:

- “Segmentation”: contains polygon’s coordinates;
- “Area”: shows the area of object;
- “IsCrowd”: shows how many objects are present in the image, ‘0’- one object, ‘1’- more than one;
- “bbox”: contains the coordinates of ground truth bounding boxes;
- “Category_id”: shows the supercategory to which the class belongs. In this case, all classes belong to the one category “animal”; “id”: unique number of each image.

Table II below provides summary statistics on the number of images of each class of developed data set. Its fragment is shown on Fig. 1.

IV. Deep Learning Approach to Detection

To solve this problem, we chose four architectures of neural networks based on the successful experience of their application for solving similar tasks [17, 18]:

1) YOLOv3 [19]: It is a one-stage neural network architecture that allows to achieve high-speed image processing with slightly lower quality. Feature extractor consists of 3x3 and 1x1 convolutional layers and shortcut connections. YOLOv3 [19] predicts boxes at 3 different scales using a similar concept to feature pyramid networks. For classification independent logistic classifier is used instead of softmax. Bounding box predictor uses anchor boxes.

TABLE II. NUMBER OF IMAGES BY CLASS

Classes	**Training sample**		**Testing sample**	
	Images	***Boxes***	***Images***	***Boxes***
Dog	4385	5508	177	218
Horse	2941	6587	128	273
Sheep	1529	9509	65	361
Cow	1968	8147	87	380
Elephant	2143	5513	89	255
Bear	960	1294	49	71
Zebra	1916	5303	85	268
Giraffe	2546	5131	101	232
Fox	460	584	10	12
Goat	274	599	14	34
Total	19122	48175	805	2104

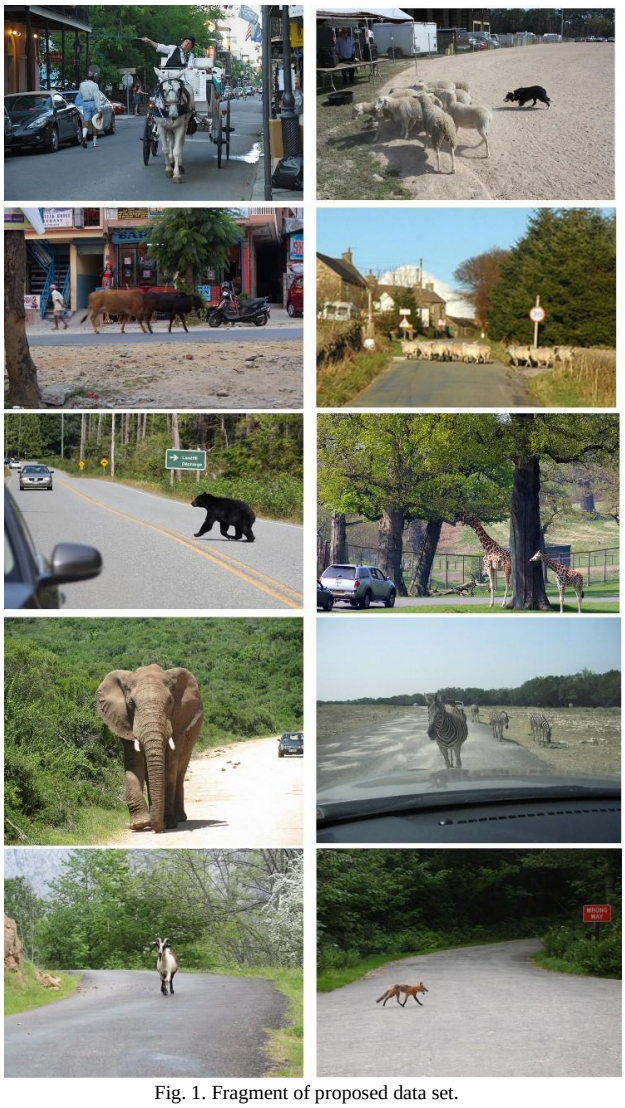

Fig. 1. Fragment of proposed data set.

2) RetinaNet R-50-FPN [20]: This one-stage network was developed to test a new loss function - the focal loss function, which was created to improve the effectiveness of training. Focal loss adds a factor $(1 - pt)\gamma$ to the standard

cross entropy criterion. Setting $\gamma > 0$ reduces the relative loss for wellclassified examples ($pt > 0.5$), putting more focus on hard, misclassified examples. The network is pretty simpe. It uses FPN (Feature pyramid network) on top of the ResNet-50 [21] architecture as feature extractor.

3) Faster R-CNN R-50-FPN [22]: This two-stage architecture uses ResNet-50 with FPN to extract feature maps. The difference between Faster R-CNN and Fast RCNN [23] is that region proposals are retrieved using the Region Proposal Network (RPN) [22] instead of using selective search which exceed network performance by about 10 times.

4) Cascade R-CNN R-50-FPN [24]: Cascade R-CNN is a multi-stage object detection architecture (Fig. 2). A specialty of this network is cascaded bounding box regression, as shown in the figure. "I" is input image, ResNet-50 with FPN is backbone, "pool" region-wise feature extraction, "H" network head, "AB" animal bounding box, and "AC" animal classification. "AB0" is proposals in all architectures.

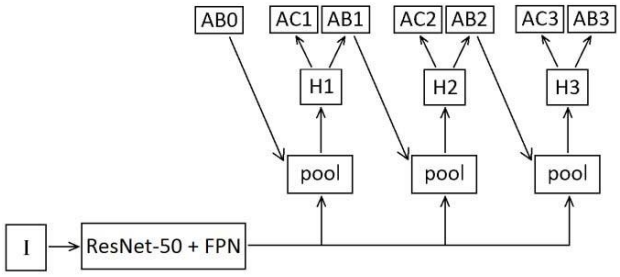

Fig. 2. Architecture of Cascade R-CNN R-50-FPN neural network

YOLOv3 was trained using the neural-network library Keras [25] (running on top of TensorFlow [26]). The rest of the architectures are using the PyTorch library [27]. For training on our data set, pre-trained models were used.

The YOLOv3 model was pre-trained on ImageNet. We used only pre-trained backbone (DarkNet53) [19]). Since we did not use the entire network, but only the backbone, the rest of the network is initialized with random weights. Because of this, during the first several epochs, the network trained with a frozen backbone to train randomly initialized weights first. Only after that the entire network is included in the training.

The remaining models were pre-trained on the COCO 2017 train [13]. Unlike YOLOv3, we used the whole pretrained network. However, since there are 80 classes in the COCO data set, before training, we removed the extra classes from the models.

The training was carried out with input image tensor sizes 640x384x3 and batch of 8 images. The learning rate was initially 0.01 and automatically decreased during the learning process if needed.

V. Experimental Results

The calculations had performed using the NVidia CUDA technology on the graphics processor of the Tesla V100 graphics card with 32GB, central processor Intel Xeon Gold 6154 CPU, 16 Core with 3.00 GHz and 128 GB RAM.

Table III shows the results of the animal detection and classification on test samples using YOLOv3, RetinaNet, Faster R-CNN and Cascade R-CNN architectures.

TABLE III. Quality of big animal detection on testing sample (10 classes)

Quality metric	Architecture of deep neural network			
	Cascade R-CNN R-50-FPN	***Faster R-CNN R-50-FPN***	***RetinaNet R-50*-FPN**	***YOLOv3***
AP_{dog}	0.81	0.81	0.83	**0.92**
AP_{horse}	0.75	0.76	0.77	**0.88**
AP_{sheep}	0.68	0.67	0.65	**0.75**
AP_{cow}	0.65	0.66	0.60	**0.80**
$AP_{elephant}$	0.82	0.83	0.84	**0.88**
AP_{bear}	0.81	0.87	0.89	**0.95**
AP_{zebra}	0.84	0.88	0.88	**0.91**
$AP_{giraffe}$	0.87	0.86	0.87	**0.91**
AP_{fox}	**0.21**	0.18	0.19	0.18
AP_{goat}	0.39	0.44	0.41	**0.58**
mAP	0.68	0.70	0.69	**0.78**

As we can see from the table above, the YOLOv3 network has the best mAP score. As for the AP in each category, YOLOv3 is slightly inferior to the Cascade R-CNN network only in the fox class. In all other classes, YOLOv3 is noticeably ahead of other architectures. The rest of the architectures showed roughly the same results.

RetinaNet has the highest speed (Table V). The slowest architecture is the Cascade R-CNN.

We had also trained models for detecting animals as one joint class, that is, without classification. The quality of detection is presented in the Table IV.

TABLE IV. Quality of big animal detection on testing sample (one joint class)

Quality metric	Architecture of deep neural network			
	Cascade R-CNN R-50-FPN	***Faster R-CNN R-50-FPN***	***RetinaNet R-50*-FPN**	***YOLOv3***
mAP	0.81	0.81	0.83	**0.92**

When detecting without classification, the mAP is higher. YOLOv3 has the best result. The rest of the architecture is about the same level. Table V shows Fps (frame per second) performance metric for the architectures providing joint class detection.

We can see that the speed has increased slightly in comparison of 10 classes detection. RetinaNet has the highest speed. The slowest architecture is Cascade R-CNN.

TABLE V. Performance of big animal detection

Performance metric	Neural network architecture

	Cascade R-CNN R-50-FPN	*Faster R-CNN R-50-FPN*	*RetinaNet R-50-FPN*	*YOLOv3*
Fps for one joint class detection	27.5	40.9	**50.0**	39.8
Fps for 10 classes detection	26.8	39.6	**44.6**	35.4

VI. Conclusion

The paper demonstrates research of deep learning approaches to detect 10 classes of big animals on the data set with about 20000 images: “Bear”, “Fox”, “Dog”, “Horse”, “Goat”, “Sheep”, “Cow”, “Zebra”, “Elephant”, “Giraffe”. Authors trained and tested several modern neural network architectures: YOLOv3, RetinaNet R-50-FPN, Faster RCNN R-50-FPN, Cascade R-CNN R-50-FPN. To compare the approaches the mAP metric was determined at IoU≥50%, also their speed was calculated for input tensor sizes 640x384x3. The highest quality metrics are demonstrated by architecture YOLOv3 as for ten classes (0.78 mAP) and one joint class (0.92 mAP) detection with speed more 35 fps on NVidia Tesla V-100 32GB video card. At the same hardware, the RetinaNet R-50-FPN architecture provided recognition speed of more than 44 fps and a 13% lower mAP. The proposed data set and neural network approach to recognizing big animals on images have shown their effectiveness and can be used in the on-board vision systems of driverless cars or in driver assistant systems.

For further study of this topic, it is necessary to increase the volume of training and testing samples for all classes especially for night and poorly lit road scenes. This can be done, for example, by using image augmentation or by the usual addition of new labeled images.

Acknowledgment

This study was carried out under the contract with the Scientific-Design Bureau of Computing Systems (SDB CS) and supported by the Government of the Russian Federation (Agreement No 075-02-2019-967).

References

[1] W. Saad, A. Alsayyari, Loose Animal-Vehicle Accidents Mitigation: Vision and Challenges. 2019 International Conference on Innovative Trends in Computer Engineering (ITCE), 2019.

[2] D. Zhou, “Real-time animal detection system for intelligent vehicles,” 2014.

[3] A. Mammeri, D. Zhou, A. Boukerche, “Animal-Vehicle Collision Mitigation System for Automated Vehicles,” IEEE Transactions on Systems, Man, and Cybernetics: Systems, Vol. 46, Iss. 9, 2016, pp. 1287-1299.

[4] G. K. Verma, P. Gupta, “Wild Animal Detection Using Deep Convolutional Neural Networks,” Second International Conference on Computer Vision & Image Processing (CVIP-2017), vol. 704, 2017.

[5] Z. Zhang, Z. He, G. Cao, W. Cao. “Animal Detection From Highly Cluttered Natural Scenes Using Spatiotemporal Object Region Proposals and Patch Verification,” IEEE Transactions on Multimedia, Vol. 18, Iss. 10 , 2016, pp. 2079-2092.

[6] K. Saleh, M. Hossny, S. Nahavandi. “Kangaroo Vehicle Collision Detection Using Deep Semantic Segmentation Convolutional Neural Network,” 2016 International Conference on Digital Image Computing: Techniques and Applications (DICTA), 2016.

[7] B. Kellenberger, M. Volpi, D. Tula, “Fast animal detection in UAV images using convolutional neural networks,” IGARSS 2017 - 2017 IEEE International Geoscience and Remote Sensing Symposium, 2017.

[8] S. Beery, D. Morris, and P. Perona, “The iWildCam 2019 Challenge Dataset,” arXiv:1907.07617, 2019.

[9] Animal Image Dataset (DOG, CAT and PANDA), https://www.kaggle.com/ashishsaxena2209/animal-image-datasetdogcat-and-panda.

[10] O. M. Parkhi, A. Vedaldi, A. Zisserman, C. V. Jawahar. Cats and Dogs. IEEE Conference on Computer Vision and Pattern Recognition, 2012

[11] A. Coates, H. Lee, and A. Y. Ng, “An Analysis of Single Layer Networks in Unsupervised Feature Learning,” AISTATS, 2011.

[12] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and F. Li, “ImageNet: A large-scale hierarchical image database,” CVPR, pp. 248-255, 2009 [13] COCO. Common objects in context, http://cocodataset.org.

[14] A. Kuznetsova, H. Rom, N. Alldrin, J. Uijlings, I. Krasin, J. PontTuset, S. Kamali, S. Popov, M. Malloci, T. Duerig, and V. Ferrari, “The Open Images Dataset V4: Unified image classification, object detection, and visual relationship detection at scale,” arXiv:1811.00982, 2018.

[15] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, “The PASCAL Visual Object Classes (VOC) Challenge,” International Journal of Computer Vision, 88(2), 303-338, 2010.

[16] S.M. Beitzel, E.C. Jensen, O. Frieder, “MAP,” In: LIU L., ÖZSU M.T. (eds) Encyclopedia of Database Systems. Springer, Boston, MA, 2009.

[17] D. A. Yudin, A. Skrynnik, A. Krishtopik, I. Belkin, A. I. Panov, “Object Detection with Deep Neural Networks for Reinforcement Learning in the Task of Autonomous Vehicles Path Planning at the Intersection,” Optical Memory & Neural Networks (Information Optics), Vol. 28 № 4, 2019.

[18] D. Yudin, A. Ivanov, M.Shchendrygin, “Detection of a Human Head on a Low-Quality Image and its Software Implementation,” International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, vol. 42, 2/W12, 2019.

[19] J. Redmon, A. Farhadi, “YOLOv3: An Incremental Improvement,” arXiv:1804.02767, 2018.

[20] T.-Y. Lin, P. Goyal, R. Girshick, K. He, P. Dollar, “Focal Loss for Dense Object Detection,” The IEEE International Conference on Computer Vision (ICCV), 2017, pp. 2980-2988.

[21] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” arXiv:1512.03385, 2015.

[22] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards RealTime Object Detection with Region Proposal Networks,” Neural Information Processing Systems, 2015.

[23] R. Girshick, “Fast R-CNN,” The IEEE International Conference on Computer Vision (ICCV), 2015, pp. 1440-1448.

[24] Z. Cai, N. Vasconcelos, “Cascade R-CNN: Delving Into High Quality Object Detection,” The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 6154-6162.

[25] Keras library, https://github.com/keras-team/keras.

[26] Tensorflow library. https://github.com/tensorflow/tensorflow.

[27] PyTorch library. https://github.com/pytorch/pytorch.