# Detection of toy soldiers taken from a bird's perspective using convolutional neural networks

Saša Sambolek and Marina Ivašić-Kos

University of Rijeka Department of Informatics, Rijeka, Croatia
sasa.sambolek@gmail.com, marinai@inf.uniri.hr

**Abstract.** This paper describes the use of two different deep-learning approaches for object detection to recognize a toy soldier. We use recordings of toy soldiers in different poses under different scenarios to simulate appearance of persons on footage taken by drones. Recordings from a bird's eye view are today widely used in the search for missing persons in non-urban areas, border control, animal movement control, and the like. We have compared the single-shot multi-box detector (SSD) with the MobileNet or Inception V2 as a backbone, SSDLite with MobileNet and Faster R-CNN combined with Inception V2 and ResNet50. The results show that Faster R-CNN detects small object such as toy soldiers more successfully than SSD, and the training time of Faster R-CNN is much shorter than that of SSD.

**Keywords:** Object detectors, SSD, Faster R-CNN

## 1 Introduction

Convolutional neural network (CNN) [21] is a particular architecture of artificial neural networks, proposed by Yann LeCun in 1988. The key idea of CNN is that the local information in the image is important for understanding the content of the image so a filter is used when learning the model, focusing on the image, part by part, as a magnifying glass. The practical advantage of such approach is that CNN uses fewer parameters than fully-connected neural networks, which significantly improves learning time and reduces the amount of data needed to train the model.

Recently, after AlexNet[22] popularized deep neural networks by winning ImageNet competitions, convolutional neuronal networks have become the most popular model for image classification and object detection problems. Image classification predicts the existence of a class in a given image based on a model that is learned on a set of labeled images. There are several challenges associated with this task, including differences between objects of the same class, similarities between objects of different classes, object occlusions, different object sizes, various backgrounds. The appearance of an object on the image might change due to lighting conditions, position (height, angle) of the camera and distance from the camera and similar [19]. The detection of an object beside the prediction of the class to which the object belongs, provides information about its location in the image, so the challenge is to solve both the classification and location task. The detected object is most often labeled with the bounding box [23], but there are also detectors that segment objects at the pixel level and mark the object using its silhouette or shape [14, 5].

Some of today's most widely used deep convolution neural networks are Faster R-CNN, RFCN, SSD, Yolo, RetinaNet. These networks are unavoidable in tasks such as image classification [22] and object detection [26], analysis of sports scenes and activities of athletes [6], disease surveillance [25], surveillance and detection of suspicious behavior [2019], describing images [17], development and management of autonomous vehicles in robotics [10], and the like.

In this paper, we have focused on the problem of detecting small objects on footage taken by the camera of a mobile device or drones from a bird's eye view. These footages are today widely used when searching for missing persons in non-urban areas, border control, animal movement control, and the like.

In [1], drones were used to locate missing persons in search and rescue operations. Authors have used HOG descriptors [8]. In [3] the SPOT system is described. It uses an infrared camera mounted on an Unmanned Aerial Vehicle and Faster R-CNN to detect villains and control animals in images. A modified MobileNet architecture was used in [9] for body detection and localization in the sea. Images were shot both with an optical camera and a multi-spectral camera. In [33] YOLO was used for detection of objects on images taken from the air. In [2424], three models of deep neural networks (SSD, Faster R-CNN, and RetinaNet) were analyzed for detection tasks on images collected by crewless aircraft. The authors showed that RetinaNet was faster and more accurate when detecting objects. The dependence analysis of Faster R-CNN, RFCN, and SSD speed and precision in case of running on different architectures was given in [18].

In this paper, we will approximate the problem of detecting small objects on bird-eye viewings or drone shots with the problem of detecting toy soldiers captured by the camera of a mobile device.

The rest of the paper is organized as follows: in Section II. we will present the architecture of CNN networks, ResNet50, Inception and MobileNet with Faster R-CNN and SSD localization methods that are used in our research. We have examined their performance on a custom toy soldiers dataset. The comparison of the detector performance and discussion are given in Section III. The paper ends with a conclusion and the proposal for future research.

## 2 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are adapted to solve the problems of high-dimensional inputs and inputs that have many features such as in cases of image processing and object classification and detection. The CNN network consists of a convolution layer, after which the network has been named, activation and pooling layers, and at the end is most often one or more fully connected layers.

The convolution layer refers to a mathematical operator defined over two functions with real value arguments that give a modified version of one of the two original functions. The layer takes a map of the features (or input image) that convolves with a set of learned parameters resulting in a new two-dimensional map. A set of learned parameters (weights and thresholds) are called filters or kernels. The filter is a 2D square matrix, small in size compared to the image to which it is applied (equal depths as well as the input). The filter consists of real values that represent the weights that need to be learned, such as a particular shape, color, edge in order to give the network good results.

The pooling layer is usually inserted between successive convolution layers, to reduce map resolution and increase spatial invariance - insensitivity to minor shifts (rotations, transformations) of features in the image as well as to reduce memory requirements for the implementation of the network. Along with the most commonly used methods (arithmetic mean and maximum [44]), there are several pooling methods used in CNN, such as Mixed Pooling, Lp Pooling, Stochastic Pooling, Spatial Pyramid Pooling and others [13].

The activation function propagates or stops the input value in a neuron depending on its shape. There is a broader range of neuron activation functions such as linear activation functions, jump functions, and sigmoidal functions. The jump functions and sigmoidal functions are a better choice for neural networks that perform classification while linear functions are often used in output layers where unlimited output is required. Newer architectures use activation functions behind each layer. One of the most commonly used activation functions in CNN is the ReLU (Rectified Linear Unit). In [13], the activation functions used in recent works are presented: Leaky Relu (LReLU), Parametric ReLU (PReLU), Randomized ReLU (RReLU), Exponential Linear Unit (ELU) and others.

A fully connected layer is the last layer in the network. The name of the fully connected layer indicates its configuration: all neurons in this layer are linked to all the outputs of the previous layer. Fully connected layers can be viewed as special types of convolution layers where all feature maps and all filters are 1 x 1.

Network hyperparameters are all parameters needed by the network and set before the network provides data for learning [2]. The hyper-parameters in convolution neural networks are learning rate, number of epochs, network layers, activation function, initialization weight, input pre-processing, pooling layers, error function.

Selecting the CNN network for feature extraction plays a vital role in object detection because the number of parameters and types of layers directly affect the memory, speed, and performance of the detector. In this paper, three types of network have been selected for feature extraction: ResNet50, Inception, and MobileNet.

## 2.1 ResNet

ResNet50 is a 50-layer Residual Network. There are other variants like ResNet101 and ResNet152 also [15]. The main innovation of ResNet is the skip connection. The skip connection in the Fig. 1 is labeled “identity.” It allows the network to learn the identity function that allows passing the input through the block without passing through the other weight layers. This allows stacking additional layers and building a deeper network, as well as overcoming the vanishing gradient problem by allowing network to skip through layers if it feels they are less relevant in training. Vanishing gradients often occurs in deep networks if no adjustment is performed because during backpropagation gradient gets smaller and smaller and can make learning difficult.

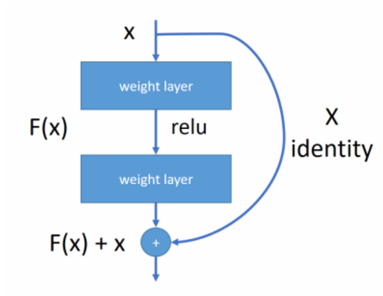

**Fig. 1.** A residual block, according to [15]

### 2.2 Inception

GoogLeNet has designed a module called Inception that approximates a sparse CNN with a normal dense construction, Fig. 2. The idea was to keep a small number of convolutional filters taking into account that only a small number of neurons are effective. The convolution filters of different sizes (5x5, 3x3, 1x1) were used to capture details on varied scales. In the versions Inception v2 and Inception v3, the authors have proposed several upgrades to increase the accuracy and reduce the computational complexity [28, 29].

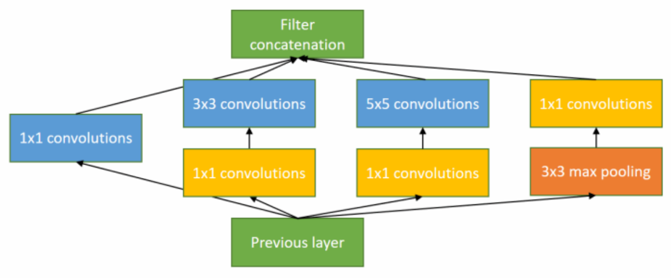

**Fig. 2.** Inception module, according to [28]

### 2.3 MobileNet

MobileNet is a lightweight architecture designed for use in a variety of mobile applications [16]. It filters the input channels by running a single convolution on each color channel instead of combining all three channels and flattening them all.

### 2.4 Faster R-CNN

In the earlier version of R-CNN [11] and Fast R-CNN [12], region proposals are generated by selective search (SS) [31] rather than using convolutional neural network (CNN). The SS algorithm was the "bottleneck" in region proposal process, so in the Faster R-CNN a separate convolution network (RPN) is used to propose regions. The RPN network then checks which location contains the object. Appropriate locations and bounding boxes are sent to the detection network that determines the class of the object and returns the bounding box of that object. This kind of design has speed up the object detection.

### 2.5 Single Shot Detector

The Single Shot Detector (SSD) method for objects detection uses deep network that omits the stage of bounding box proposal and allows features extraction without losing accuracy. The approach assumes that potential objects can be located within the predefined bounding box of different size and side ratios centered in each location of feature map. The network for each bounding box determines the probability measure for the presence of each of the possible categories and adjusts the position of the box to frame the object better. In order to overcome the problems inherent in the difference in object sizes, the network makes decisions by combining prediction from several feature maps of different dimensions [23].

## 3 Comparison of SSD and Faster RCNN detection performance on scenes of toy soldiers

We have tested and compared the accuracy of the object detector for a class of person (toy soldier) at different scene configurations, changing the number of object, their position, background complexity and lighting conditions. The goal is to select the appropriate model for future research on the detection of missing persons in rescue operations.

We used publicly available pre-trained models with corresponding weights learned on the Microsoft's common object and context (COCO) dataset [7] by transfer learning and fine-tune the model parameters on our data set.

We used the Tensorflow implementation [30] of the CNN model and the Python programming language in the Windows 10 x64 environment. All models were trained on a laptop with the i5-7300HQ CPU and the Ge-Force GTX 1050Ti 4GB GPU. The number of epochs and the training time differs among models and depends on loss. The parameters of each model have not been changed and were equal to the parameters of the original model.

## 3.1 Data Preprocessing

The data set contains 386 images shot by a mobile device camera (Samsung SM-G960F) at a 2160x2160px resolution, without using a tripod. Each image contains multiple instances of toy soldiers, taken under different angles and different lighting conditions with a different background type from a uniform to complex (such as grassy surfaces). The images are divided into a learning and test set in a cutoff of 80:20, and their resolution is reduced to 720x720px. In total, there are 798 toy soldiers in the images, of which 651 are in learning set and 147 in test.

The LabelImg tool was used to plot bounding box and create responsive XML files with stored xmin, xmax, ymin, ymax position for each layout. Images and corresponding XML files are then converted to TFRecord files that are implemented in the Tensorflow environment. TFRecord files merge all the images and notes into a single file, thus reducing the training time by eliminating the need of opening each file.

## 3.2 Methods

**SSD with MobileNet**
This method uses SSD for detection while the MobileNet network is used as a feature extractor. The output of MobileNets is processed using the SSD. We have tested the detection results of two versions of the MobileNet network (V1 and V2), referred to as ssd_mobilenet_v1 and ssd_mobilenet_v2. Both networks were pre-trained (ssd_mobilenet_v1_coco_2018_01_28 and ssd_mobilenet_v2_coco_2018_03_29) on COCO dataset of 1.5 million objects (80 categories) in 330,000 images. We trained the network using toy soldier's images width bounding box as input to the training algorithm. The network parameters include: prediction dropout probability 0.8, kernel size 1 and a box code size set to 4. The root mean square propagation optimization algorithm is used for optimizing the loss function with learning rate of 0.004 and decay factor 0.95. At the non-maximum suppression part of the network a score threshold of $1 \times 10^{-8}$ is used with an intersection of union threshold of 0.6, both the classification and localization weights are set to 1. Ssd_mobilenet_v1 was trained for 17,105 steps and ssd_mobilenet_v2 for 10,123 steps.

**SSD with Inception-V2**
The combination of SSD and Inception-V2 is called SSD-Inception-V2. In this case, SSD is used for detection while Inception-V2 extracts features. We trained the network using predefined ssd_inception_v2_coco_2018_01_28 weights. The training process uses similar hyperparameters as SSD with MobileNet, except in this case of the kernel size that is set to 3. The network was trained for 6,437 steps.

**SSDLite with MobileNet-V2**
SSDLite [27] is a mobile version of the regular SSD, so all regular convolutions with detachable convolutions are replaced (depthwise followed by $1 \times 1$ projection) in SSD layers. This design is in line with the overall design of MobileNet and is considered to

be much more efficient. Compared to SSD, it significantly reduces the number of parameters and computing costs. We trained the network using pre-trained ssdlite_mobilenet_v2_coco_2018_05_09 weights. Similar hyperparameters were used as before, and the network was trained for 14,223 steps.

**Faster R-CNN with ResNet50**

Faster R-CNN detection involves two phases. The first phase requires a region proposal network (RPN) that allows simultaneous prediction of object anchors and confidence (objectiveness) from some internal layers. For this purpose, a residual network with a depth of 50 layers (ResNet50) is used. The grid anchor size was 16 x 16 pixels with scales [0.25, 0.5, 1.0, 2.0], a non-maximum-suppression-IoU threshold was set to 0.7, the localization loss weight to 2.0, objectiveness weight to 1.0 with an initial crop size of 14, kernel size was 2 with strides set to 2. The second phase requires information from the first phase to predict the class label and the bounding box. We trained the network using pre-trained faster_rcnn_resnet50_coco_2018_01_28 weights. The IoU-threshold for prediction score was set to 0.6; the momentum (SGD) optimizer for optimizing the loss functions has initial learning rate set to 0.0002 and momentum value 0.9. The network was trained for 12,195 steps.

**Faster R-CNN with Inception-V2**

Faster R-CNN uses the Inception V2 feature extractor to get features from the input image. The middle layer of the Inception module uses the RPN network component to predict the object anchor and confidences. As in previous cases, the network was trained with pre-trained fast-er_rcnn_inception_v2_coco_2018_01_28 weights. Similar hyperparameters were used as in case of Faster R-CNN with ResNet50 and the learning process lasted for 33,366 steps.

## 3.3 Results and discussion

We compared the results of the SSD model and the Faster RCNN object detector based on CNNs on our toy soldiers test set concerning mean average precision (mAP) [32]. A detection is considered as true positive when more than half of the area belonging to the soldier is inside the detected bounding box. Detectors performance are also evaluated in terms of recall, precision and F1 score.

$$F1 = \frac{2 \cdot \text{Recall} \cdot \text{Precision}}{(\text{Recall} + \text{Precision})} \quad (1)$$

Fig. 3. shows a comparison of results of models that were additionally learned on our learning set with original models trained on the COCO dataset. The results show a significant increase in the average precision of all models after training on our dataset. The best results of over 96% were achieved with the faster_rcnn network. The implementation of faster_rcnn with Resnet50 proved to be somewhat successful than architecture with the Inception Network. Faster_rcnn has also shown the best classification results concerning F1 score and Recall [19], Fig. 4. All classification result of all models in terms of precision, recall and F1 score are shown in Fig. 4.

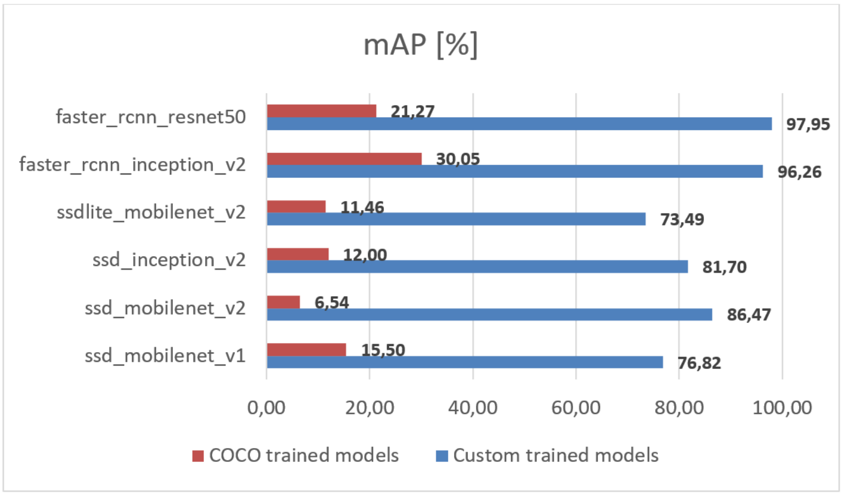

**Fig. 3.** Comparison of the evaluation result of the toy soldier's detection

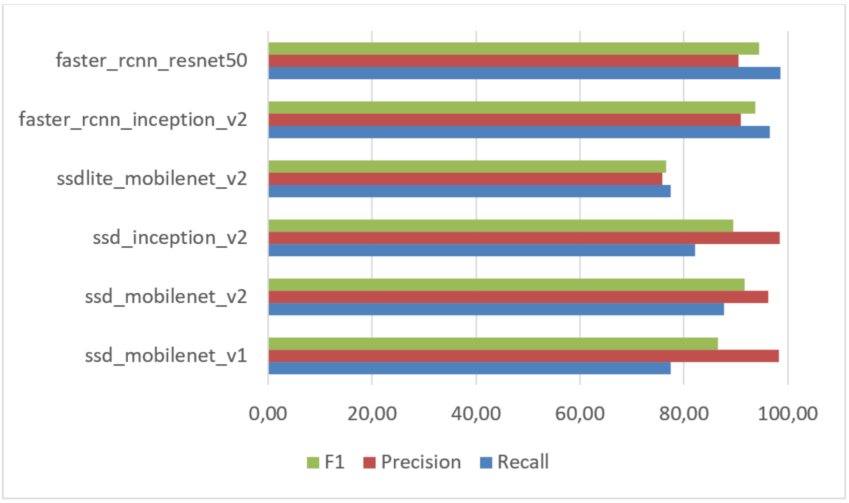

**Fig. 4.** Comparison of the results of the trained model detection concerning the F1, Precision and Recall metrics

Fig. 5. shows the time required to train the model on our learning set. The least amount of time was needed to learn the faster_rcnn model. The longest, more than 3.5 times longer than learning the fast_rcnn model, was needed to learn the ssdlitle-mobilenet_v2 model.

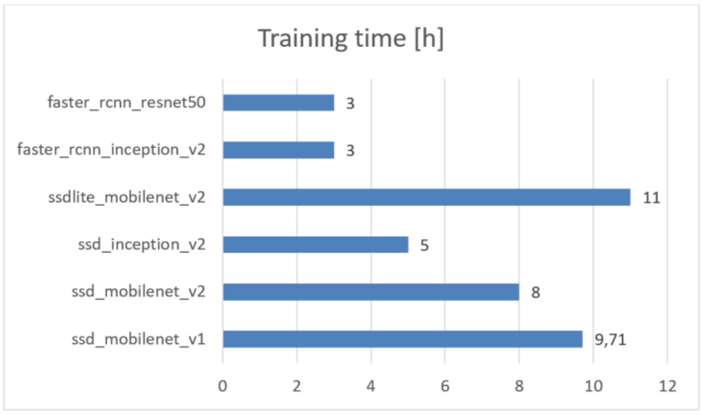

**Fig. 5.** Comparison of model learning time on the custom dataset

Model performances are additionally presented in two scenarios: simple and complex. The simple scenario has a uniform background color and up to 8 visible objects near the camera, which may overlap. A complex scenario is considered when the number of objects in a scene is equal to and greater than 9, away from the camera and with occlusions. A Fig. 6. shows an example of the detection results in the case of a simple scenario. The images marked A through F show the same scenarios with a uniform background with a wooden pattern and five soldiers in different poses such as walking with a gun, shooting, crawling and lying down.

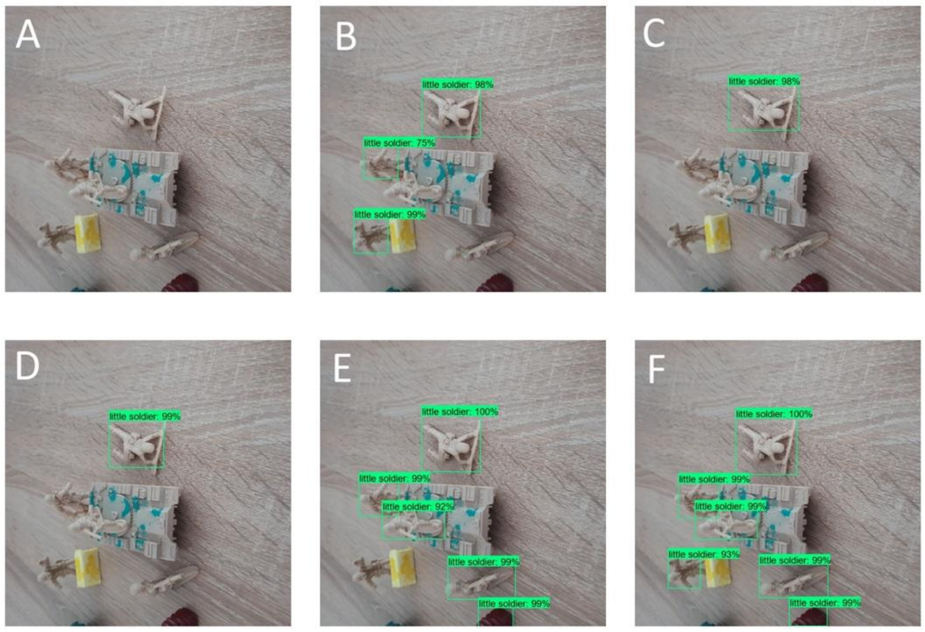

**Fig. 6.** The detection results in the case of a simple scenario

In all the images, the results of individual models are indicated in the following way:

- A – ssd_mobilenet_v1,
- B – ssd_mobilenet_v2,
- C – ssdlite_mobilenet_v2,
- D – ssd_inception_v2,
- E – faster_rcnn_resnet_50,
- F – faster_rcnn_inception_v2.

In figure A, no soldier was detected, B has 3 of 5 true positive (TP) detections, C and D only one detected soldier, while E has 4 TP with one false positive (FP) detection, and F all positive detections with one FP.

In the case of a uniform background with higher contrast to soldiers, as in Fig. 7. all models have detected with greater success in comparison to the previous case, even though in this example, we have a higher number of soldiers and at a greater distance. There were 11 soldiers on the scene, but no model detected a soldier on a tank of the identical color. The best results were achieved in E and F images with 10 successful detections, then model B with 7, and then models A and C follow. The sequence of the success of the model is similar to the one in the previous example.

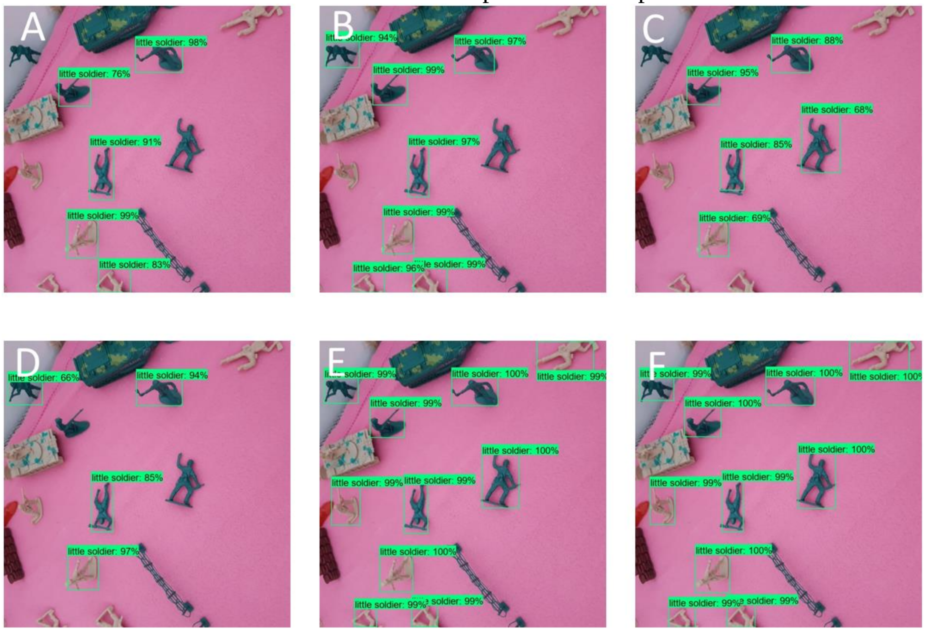

**Fig. 7.** The case of a uniform background with higher contrast to soldiers

Fig. 8. shows a complex scene in which soldiers are partially covered with grass. The camera's position is not as in the previous cases from the top, but from the side. The models in Figures A and D did not have any detection, while B detected almost the entire image as a soldier. C has one positive and two false detections, E has repeatedly detected the same object, but with different rectangle size and has a false detection, and F has an accurate, true positive detection.

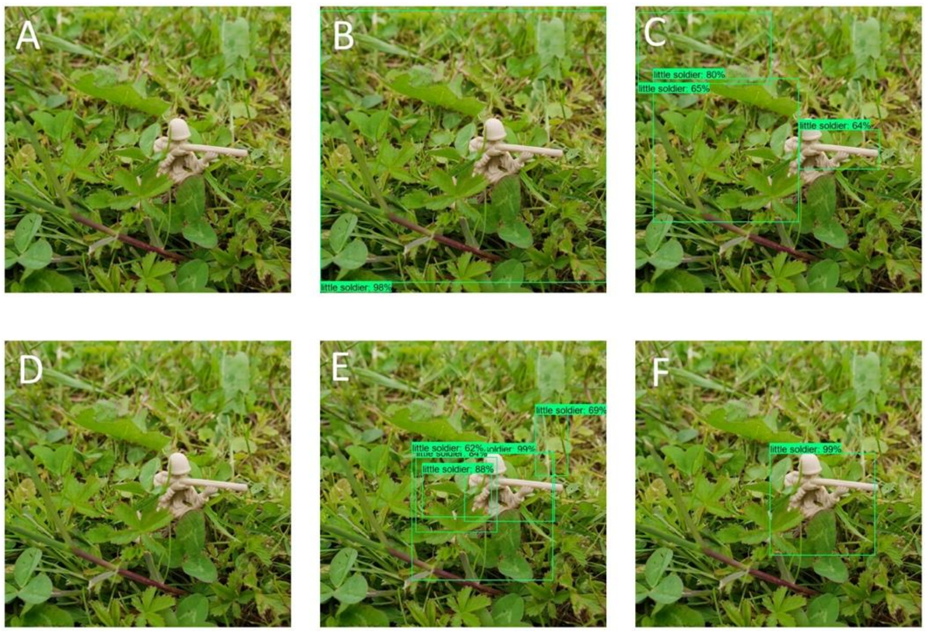

**Fig. 8.** A complex scene in which soldiers are partially covered with grass

Fig. 9. shows two scenes with a camera positioned from a bird's perspective at a greater distance than in earlier cases and with 8 soldiers on a uniform blue background. Model A detected only one soldier (Fig. 9.a) whereas B, C, and D had no detection (Fig. 9.b). Model E detected all soldiers (9.d), while F detects all soldiers plus three false detections (9.e).

In the second scene recorded from a greater distance on the grass, Fig. 9.c and 9.f, only F detects one soldier out of 7 possible. Examples show that all models have problems with object detection when an object is less than 50px in height or width, especially when the contrast of the subject and background is not significant, and when the background is more complex than in the case of grass.

Fig. 10 is an example of a scene with two soldiers with a cluttered background. E and F models detected both soldiers with a probability of detection of 100% (Fig. 10.a, Fig. 10.c, and Fig. 10.e), while model B detected only one soldier (Fig. 10.b, Fig. 10.d.) and other models failed to detect anything. Fig. 10.e shows a higher contrast between the soldiers and the background, but this did not help models A, B, C, D to have a successful detection. Fig. 10.f shows the occlusion of soldiers; however, models B, D, E, and F were able to detect them.

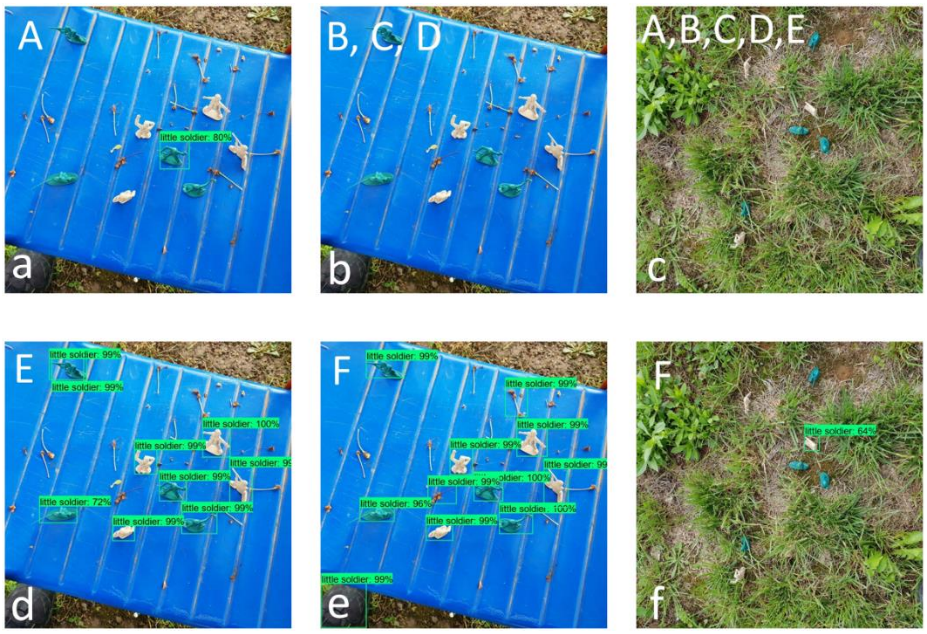

**Fig. 9.** Two scenes with a camera positioned from a bird's perspective

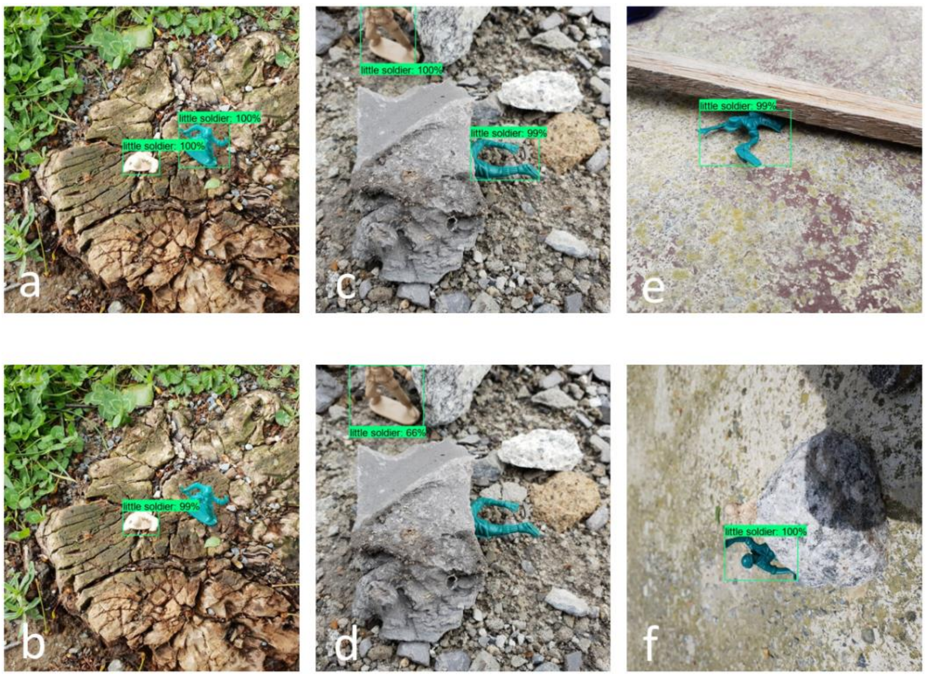

**Fig. 10.** A scene with a cluttered background and the occlusion of soldiers

## 4 Conclusion

Recordings taken from the air today are used mainly in search of missing persons, in mountain rescue, in the control of animal movement, and the like. The ability to automatically detect persons and objects on the images taken from a bird's perspective would greatly facilitate the search and rescue of people or the control of people and animals.

CNN networks have proven successful in classification and object detection tasks on general-purpose images, and in this paper, we have tested their performance in detecting toy soldiers taken from the bird's eye view. On the custom dataset, we compared the performance of ResNet50, Inception, and MobileNet networks with Faster RCNN and SSD methods of localization. The analysis of the obtained results shows that Faster RCNN is more suitable for detection because it detects toy soldiers more successfully. The configuration with the Inception network is more successful than the configuration with ResNet50. The problem with this method is that it requires more time and computation power.

The examples also show the background effect on detection accuracy. With a uniform background and higher color contrast, detection of all models is significantly successful than in case of detection at a greater distance, on the grass, and with semi-hidden objects. In future work, we will try to find a way to solve this problem.

This paper provides a promising base ground for further research in real-time detection of missing persons in search and rescue operations. We plan to investigate the further use of different detection methods (speed, accuracy) on the android system.

## Acknowledgment

This research was supported by Croatian Science Foundation under the project IP-2016-06-8345 "Automatic recognition of actions and activities in multimedia content from the sports domain" (RAASS) and by the University of Rijeka under the project number 18-222-1385.

## References

1. Andriluka, M., Schnitzspan, P., Meyer, J., Kohlbrecher, S., Petersen, K., Von Stryk, O., ... & Schiele, B.: Vision-based victim detection from unmanned aerial vehicles. IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1740-1747 (2010)
2. Bengio, Y.: Practical recommendations for gradient-based training of deep architectures. In Neural networks: Tricks of the trade pp. 437-478 (2012)
3. Bondi, E., Fang, F., Hamilton, M., Kar, D., Dmello, D., Choi, J., ... & Nevatia, R.: Spot poachers in action: Augmenting conservation drones with automatic detection in near real-time. In Thirty-Second AAAI Conference on Artificial Intelligence. (2018)
4. Boureau, Y., Ponce, J., & LeCun, Y.: A theoretical analysis of feature pooling in vision algorithms. In Proc. International Conference on Machine learning (Vol. 28), (2010)

5. Burić, M., Pobar, M., Ivašić-Kos, M.: Ball Detection using Yolo and Mask R-CNN, In 2018 International Conference on Computational Science and Computational Intelligence (CSCI'18), pp. 319-323, Las Vegas (2018)
6. Burić, M., Pobar, M., Ivašić-Kos, M.: Adapting YOLO Network for Ball and Player Detection, Proceedings of the 8th International Conference on Pattern Recognition Applications and Methods - Volume 1: ICPRAM, SciTePress, pp. 845-851, Prag (2019)
7. Cocodataset, http://cocodataset.org/, last accessed 2019/5/25
8. Dalal, N., & Triggs, B.: Histograms of oriented gradients for human detection. In international Conference on computer vision & Pattern Recognition, pp. 886-893 (2005)
9. Gallego, A. J., Pertusa, A., Gil, P., & Fisher, R. B.: Detection of bodies in maritime rescue operations using unmanned aerial vehicles with multispectral cameras. Journal of Field Robotics.
10. Gao, H., Cheng, B., Wang, J., Li, K., Zhao, J., and Li, D.: Object Classification Using CNN-Based Fusion of Vision and LIDAR in Autonomous Vehicle Environment, in IEEE Transactions on Industrial Informatics, vol. 14, no. 9, pp. 4224-4231 (2018)
11. Girshick, R., Donahue, J., Darrell, T., & Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 580-587 (2014)
12. Girshick, R.: Fast R-CNN. In Proceedings of the IEEE international conference on computer vision, pp. 1440-1448 (2015)
13. Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., ... & Chen, T.: Recent advances in convolutional neural networks. Pattern Recognition, 77, pp. 354-377 (2018)
14. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask R-CNN, 2017 IEEE International Conference on Computer Vision (ICCV), pp. 2980-2988, Venice (2017)
15. He, K., Zhang, X., Ren, S., & Sun, J.: Deep residual learning for image recognition. In Proc. of the IEEE conference on computer vision and pattern recognition, pp. 770-778 (2016).
16. Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., ... & Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications, (2017)
17. Hrga, I., Ivašić-Kos, M.: Deep Image Captioning: An Overview, In IEEE MIPRO, Opatija (2019)
18. Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., Murphy, K.: Speed/accuracy trade-offs for modern convolutional object detectors. IEEE Conference on computer vision and pattern recognition, pp. 7310-7311 (2017)
19. Ivašić-Kos, M., Ipšić, I., Ribarić, S.: A knowledge-based multi-layered image annotation system. Expert systems with applications. 42, pp. 9539-9553 (2015)
20. Ivasic-Kos, M., Kristo, M., Pobar, M.: Human detection in thermal imaging using YOLO, In ICCTA 2019, Istanbul (2019)
21. Johnson, J., Karpathy, A.: Convolutional Neural Networks, Stanford Computer Science, https://cs231n.github.io/convolutional-networks, last access 2019/3/11.
22. Krizhevsky, A., Sutskever, I., Hinton, G. E.: Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems, pp. 1097-1105 (2012)
23. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C.: Ssd: Single shot multi-box detector. In European conference on computer vision, pp. 21-37 (2016)
24. Radovic, M., Adarkwa, O., Wang, Q.: Object recognition in aerial images using convolutional neural networks. Journal of Imaging (2017)

25. Ramcharan, A., McCloskey, P., Baranowski, K., Mbilinyi, N., Mrisho, L., Ndalahwa, M., & Hughes, D.: Assessing a mobile-based deep learning model for plant disease surveillance (2018)
26. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-CNN: Towards real-time object detection with region proposal networks. In Advances in neural information processing systems, pp. 91-99 (2015)
27. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L. C.: Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4510-4520 (2018)
28. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Rabinovich, A.: Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1-9 (2015)
29. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z.: Rethinking the inception architecture for computer vision. In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 2818-2826 (2016)
30. Tensorflow object detection models zoo, https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md, last accessed 2019/5/25
31. Uijlings, J. R., Van De Sande, K. E., Gevers, T., & Smeulders, A. W.: Selective search for object recognition. International journal of computer vision,104(2), 154-171 (2013)
32. Visual Object Classes Challenge 2012 (VOC2012), http://host.robots.ox.ac.uk/pascal/VOC/voc2012/, last accessed 2019/5/25
33. Wang, X., Cheng, P., Liu, X., & Uzochukwu, B.: Fast and Accurate, Convolutional Neural Network Based Approach for Object Detection from UAV. In IECON 2018-44th Annual Conference of the IEEE Industrial Electronics Society, pp. 3171-3175 (2018)