Intelligent Automation & Soft Computing
DOI: 10.32604/iasc.2023.032526
Article

Tech Science Press

Deep Transfer Learning Approach for Robust Hand Detection

Stevica Cvetkovic[1,*], Nemanja Savic[1] and Ivan Ciric[2]

[1]Faculty of Electronic Engineering, University of Nis, Nis, 18000, Serbia
[2]Faculty of Mechanical Engineering, University of Nis, Nis, 18000, Serbia
*Corresponding Author: Stevica Cvetkovic. Email: stevica.cvetkovic@elfak.ni.ac.rs
Received: 20 May 2022; Accepted: 06 July 2022

Abstract: Human hand detection in uncontrolled environments is a challenging visual recognition task due to numerous variations of hand poses and background image clutter. To achieve highly accurate results as well as provide real-time execution, we proposed a deep transfer learning approach over the state-of-the-art deep learning object detector. Our method, denoted as YOLOHANDS, is built on top of the You Only Look Once (YOLO) deep learning architecture, which is modified to adapt to the single class hand detection task. The model transfer is performed by modifying the higher convolutional layers including the last fully connected layer, while initializing lower non-modified layers with the generic pre-trained weights. To address robustness issues, we introduced a comprehensive augmentation procedure over the training image dataset, specifically adapted for the hand detection problem. Experimental evaluation of the proposed method, which is performed on a challenging public dataset, has demonstrated highly accurate results, comparable to the state-of-the-art methods.

Keywords: Deep learning model; object detection; hand detection; transfer learning; data augmentation

1 Introduction

Accurate detection of human hands in images is of crucial importance for many high-level human behavior analysis tasks. A variety of applications in robotics and human-computer interaction, such as virtual reality [1], driver behavior monitoring [2] or sign language recognition [3,4]; heavily rely on accurate results from hand detection methods. It is still an extremely challenging task as hands can vary in shape and viewpoint, can be partially closed or occluded, can have different articulations of the fingers, etc.

Recent expansion of Deep Neural Networks (DNN) has led to adoption of this approach to the visual object detection task, which significantly improved the performance of conventional object detection methods. Beside significant detection accuracy improvements brought by modern DNN-based methods [5,6], there has been advances in terms of execution speed [7–9]. However, there is still a lack of research in terms of extension and adaptation of these methods to the specific domain problems, such as human hand detection in images. A powerful technique for adapting and reutilizing a generic pre-trained DNN on a specific domain task with a smaller target dataset, is denoted as Transfer Learning [10–13].

There are two main reasons for adapting a pre-trained DNN, instead of building and training a completely new network model from scratch. First, when the target dataset is significantly smaller than the original generic dataset, transfer learning enables additional training of large networks without overfitting. Also, it is an enormously faster approach than complete retraining, which is especially the case for large DNN models which took days or even months to be trained on large public datasets using advanced Graphics Processing Unit (GPU) architectures.

Therefore, in this paper, we investigated how to successfully apply a transfer learning approach over state-of-the-art DNN-based object detectors, in order to make them robust for a specific task of human hand detection. The main contributions of the paper are summarized as follows:

- We transferred and trained a Deep Neural Network based on a YOLO network architecture [9], by modifying it to adapt to the single class hand detection task.
- We proposed a comprehensive data augmentation procedure over the training image dataset, in order to address robustness issues.
- Training and evaluation of the proposed method, which is performed on a challenging public dataset, has demonstrated highly accurate results, comparable to the state-of-the-art methods.

In the rest of the paper we will first give an overview of related work, followed by a short description of the original YOLO object detection method. Then, we will present a detailed description of our approach for transferring the YOLO model to a robust hand detection method, denoted as YOLOHANDS. Description of the proposed data augmentation procedure is given before presenting the results of comprehensive evaluation on a publicly available dataset.

2 Related Work

Many conventional computer vision methods have been proposed in the literature to detect human hands in color images. One of the first successful methods was proposed in [14]. It generates skin-based region proposals followed by a sliding window shape-based detector to increase detection recall. However, methods relying on skin detection have serious limitations in environments with poor illumination. Furthermore, they could be ambiguous if face regions are also present in the image. First applications of Deep Neural Networks (DNN) for object detection tasks started as extensions to the standard classification methods. They relied on external region proposal step to identify object region candidates which were afterwards classified using the DNN.

2.1 Region-Proposal Deep Learning Methods for Object Detection

One of the first successful DNN methods for object detection was introduced in [6]. It starts with a segmentation algorithm called "selective search", to preselect large number of object region candidates (approx. 2000 candidates). Then, it applies a pretrained DNN to extract a fixed-length feature vector from each region proposal, which is followed by a Support Vector Machine (SVM) multi-class classifier. Slow execution times and limited accuracy were serious drawbacks of that method. Some improvements to the original method were presented in [5]. Instead of performing over 2000 forward passes for each object, it computes a convolutional feature map for the complete image in a single forward pass, making it significantly faster. Another improvement is that the neural network is trained end-to-end with a multi-task loss, resulting in a simpler training procedure. However, the speed of the method was still far from real-time performances. In [7], a significant improvement of the speed was presented, by abandoning the traditional region proposal method, and relying on a fully deep learning approach. The proposed network architecture consists of the two sub-networks which are merged into a single network and trained end-to-end. The method, denoted as Faster Region-based Convolutional Neural Network (Faster R-CNN) is still considered as one of the optimal object detectors in terms of accuracy [15].

2.2 Single-Stage Deep Learning Methods for Object Detection

While region-proposal methods might achieve higher accuracy, they still have significant drawbacks in terms of computation complexity. Single-stage object detectors use a single deep neural network to simultaneously predict object's bounding box and class probabilities. This allows them to have significantly faster inference times, while still achieving the high accuracy of detections. The two most notable representatives of single-stage methods are Single Shot Detector (SSD) [16] and YOLO [8], where YOLO has demonstrated higher speed and accuracy of results. In this study we will rely on YOLOv3 Deep Neural Network architecture [9] for developing our robust hand detection method. We will first give a brief overview of a basic YOLO network architecture, followed by details of our modifications and adaptations to the specific problem of human hand detection in unconstrained environments.

2.3 Transformers for Object Detection

Transformer architecture [17], which has had a tremendous impact in the Natural Language Processing (NLP) domain, has recently been introduced into computer vision applications. One of the first successful methods to demonstrate how Transformers can replace standard convolutions in deep neural networks on large image datasets, was introduced in [18]. Vision Transformers (ViT) applied the original Transformer model on a sequence of image patches flattened as vectors. It has gained significant attention, and a number of recent methods have been proposed which build upon ViTs.

Object detection methods relying on Transformer modules, could be grouped into the following three categories. One group of methods use transformer backbones for feature extraction, combined with R-CNN head for detection [19]. Another group uses CNN backbone for visual feature extraction, followed by a Transformer based decoder for object detection [20]. Finally, an entirely transformer-based method for object detection is introduced in [21]. Since the transformer-based methods achieve state-of-the-art results, their potential to replace CNN networks for object detection tasks is very realistic.

3 Transferring a Generic Object Detection Model

We rely on a state-of-the-art deep neural network architecture for object detection–YOLOv3 [9], as a basis for developing our hand detection network. To adapt the network to the hand detection task, we applied a transfer learning approach [11,22–24].

3.1 Generic Object Detection Using YOLO Method

The YOLO object detection method was first introduced in 2016 [25]. The method is based on a single deep neural network trained to take an image as input and predict bounding boxes, class labels, and confidence scores for each detected bounding box. The initial version of YOLO neural network was operating very fast at 155 fps, while producing a relatively lower accuracy compared to the previously described region-proposal methods. Its algorithm starts by splitting the input image into $S \times S$ grid cells, where each cell is responsible for detecting objects whose centroid fall into that grid cell. Concretely, each grid cell can predict (regress) up to B bounding boxes, and for each of them it outputs the confidence p that the box contains the object. This means that each grid cell gives B outputs consisting of a 5-element vector $[x, y, w, h, p]$, where x, y represent center of the object's bounding box relative to the bounds of the grid cell; w, h represent width and height relative to the whole image dimensions, and p is detectors confidence (probability) that the object exists within that bounding box. In addition, for each grid cell, YOLO outputs a list of C-element class probabilities represented as one hot encoded vector. Finally, the neural network output is given as a vector of $S \times S \times B \times (5 + C)$ elements. In the case of a 416×416 input image, a grid of 13×13 cells can be used with $B = 3$ bounding boxes predicted for each cell, and $C = 80$ object classes in case of the COCO dataset [26]. It will finally produce an output vector of $13 \times 13 \times 3 \times (5 + 80) = 43095$ elements for an input image.

There have been many improvements of the initial YOLO detector over the years [8,9,27]. The improved YOLOv3 introduced a couple of significant improvements [9]. To create a network which is resilient to object size variations, the output vector is composed of three different output layers that take feature maps produced at different stages in the base architecture. This allows those layers to detect objects at three separate scales, inspired by work done in [16]. Since YOLOv3 considers three different scales, the main difference in outputs will be reflected in varying grid sizes. Instead of considering a single grid of 13×13 cells ($S = 13$) for an input image of 416×416 pixels, the parameter S will change depending on the scale, so it will finally generate three different grid sizes: 13×13, 26×26 and 52×52. It means that the dimension of the output vector of the YOLOv3 network, when taking all three scales into account, will be of size: $13 \times 13 \times B \times (5 + C) + 26 \times 26 \times B \times (5 + C) + 52 \times 52 \times B \times (5 + C)$. The base network architecture that determines convolutional feature maps has also been slightly changed. It now contains 53 layers, composed of 3×3 and 1×1 convolutional filters. Also, residual blocks with skip connections have been added to battle vanishing gradients, inspired by [28]. An illustration of a network output for $C = 1$ can be seen in Fig. 1 below.

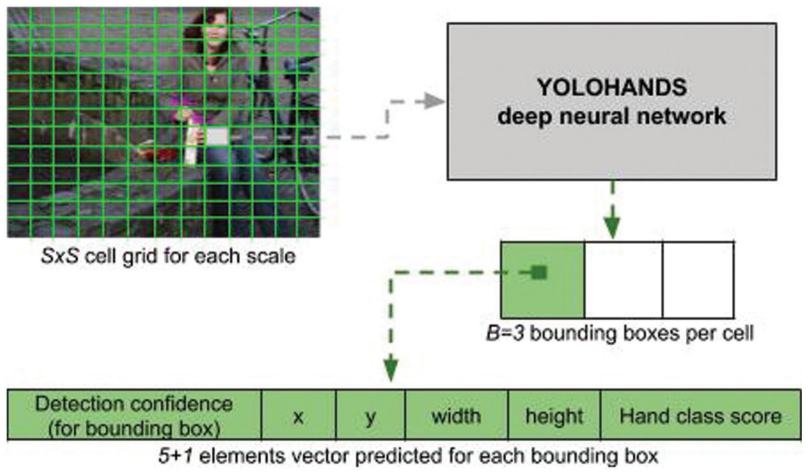
YOLOHANDS deep neural network
SxS cell grid for each scale
B=3 bounding boxes per cell
Detection confidence (for bounding box)
x
y
width
height
Hand class score
5+1 elements vector predicted for each bounding box

Figure 1: Illustration of the detection output for one image scale (3 scales are used)

3.2 A Deep Transfer Learning Approach for Robust Hand Detection

Our approach relies on the YOLOv3 network, which was originally created to detect $C = 80$ distinct object classes that are present in the publicly available COCO dataset [26]. Since we only have an interest in detecting a single class (i.e., human hand), we need to modify certain layers of the network, including the final fully connected layers and the convolutional layers preceding them (for each of the three scales). Therefore, the last fully connected layer "FC1" with $S \times S \times B \times (5 + C) = 13 \times 13 \times 3 \times (5 + 80) = 43095$ outputs in the original YOLOv3 network, was replaced with a fully connected layer with $13 \times 13 \times 3 \times (5 + 1) = 3042$ outputs in our YOLOHANDS. The same modification has been applied to the other two fully connected layers ("FC2" and "FC3", with grid sizes of $S = 26$ and $S = 52$, respectively). The reduction in the size of the fully connected layer was mirrored by the number of filters in the preceding convolutional layer. The formula used to compute the number of output filters in the preceding convolutional layers is $B \times (5 + C) = 3 \times (5 + 1) = 18$, so the number of filters is also reduced to 18 (instead original 255).

For initializing weights of the YOLOHANDS neural network, there are generally two options. Either to randomly initialize the weights in the network and start training from scratch, or to initialize the weights with previously learned values on some larger generic dataset. Since our YOLOHANDS model is built by modifying the higher convolutional layers including the last fully connected layer, we decided to initialize weights of lower non-modified convolutional layers with the existing pretrained weights, and to use random initialization for the introduced modified layers. There are several publicly available pretrained weight sets for YOLOv3 which are learned on several large public datasets, but two of them are the most commonly used: "yolov3.weights" and "darknet53.conv.74.weights" [9]. The first one, "yolov3.weights" contains weights for all the layers in the network that are learned on the COCO dataset [26], and the other one "darknet53.conv.74.weights" contains weights only for lower convolutional layers of the network that are learned on a larger ImageNet dataset [29]. Since the later one is trained on a larger dataset, we used that one for our model. The overall architecture of our approach is given in Fig. 2 below.

When choosing the training procedure for the network weights tuning, there are few options. One option is to lock the first several layers during training, meaning that the locked layers won't be updated with gradients during backpropagation. In this case we would only update the weights of the last few layers. Although this can significantly reduce training speed and memory consumption, it is only applicable to detect objects with a similar appearance to the generic objects used for the original YOLO training. However, we are detecting a specific object class (i.e., human hands) which does not exist as a separate class in the training dataset of the original YOLO detector, so we decided to perform the training over all network layers. In this way we open all layers for adaptation during the backpropagation procedure.

Training of the YOLOHANDS model is performed in iterations, where each iteration consists of a single forward-backward pass over one batch of images. We used batches consisting of 64 images, due to GPU memory limitations. After every 1000 iterations, current weights of the network are stored to serve as checkpoints.

The original network outputs regressed bounding boxes around potential object locations, coupled with a detection confidence of the bounding boxes. By default, regressed bounding boxes whose confidence is below the value 0.5 are discarded. The option to change the confidence threshold values is exposed as a parameter of the network, and we take advantage of that for computing the Average Precision (AP) measure during the evaluation.

4 Data Preprocessing

Choosing the appropriate dataset, as well as defining the corresponding preprocessing procedure, might be of crucial importance for Deep Learning applications. Therefore, we will first describe the dataset and give details of the proposed data augmentation algorithm.

4.1 Dataset Description

The dataset used for evaluation is a publicly available Oxford Hand Dataset [14]. It has been created by aggregating several different image sources and annotated by a bounding box rotated with respect to the wrist orientation. All hand instances were split into two main groups: big and normal hand instances. Big hand instances are those that are larger than 1500 square pixels. For training and evaluation purposes in this study, two subsets of the Oxford Hand Dataset have been used–a training subset consisting of 9163 hand instances and a test subset of 2031 instances for evaluation. We were using the complete dataset, without making any distinction between 'normal' and 'big' hand instances. The original annotations for the Oxford Hand Dataset had to be slightly modified, since the architecture of the original YOLO detector is not designed for oriented object detection. Actually, YOLO network only accepts axis-aligned bounding boxes for training and produces axis-aligned bounding boxes during inference. Therefore, for every original 'rotated' annotation box we had to compute an axis-aligned bounding box that completely covers it.

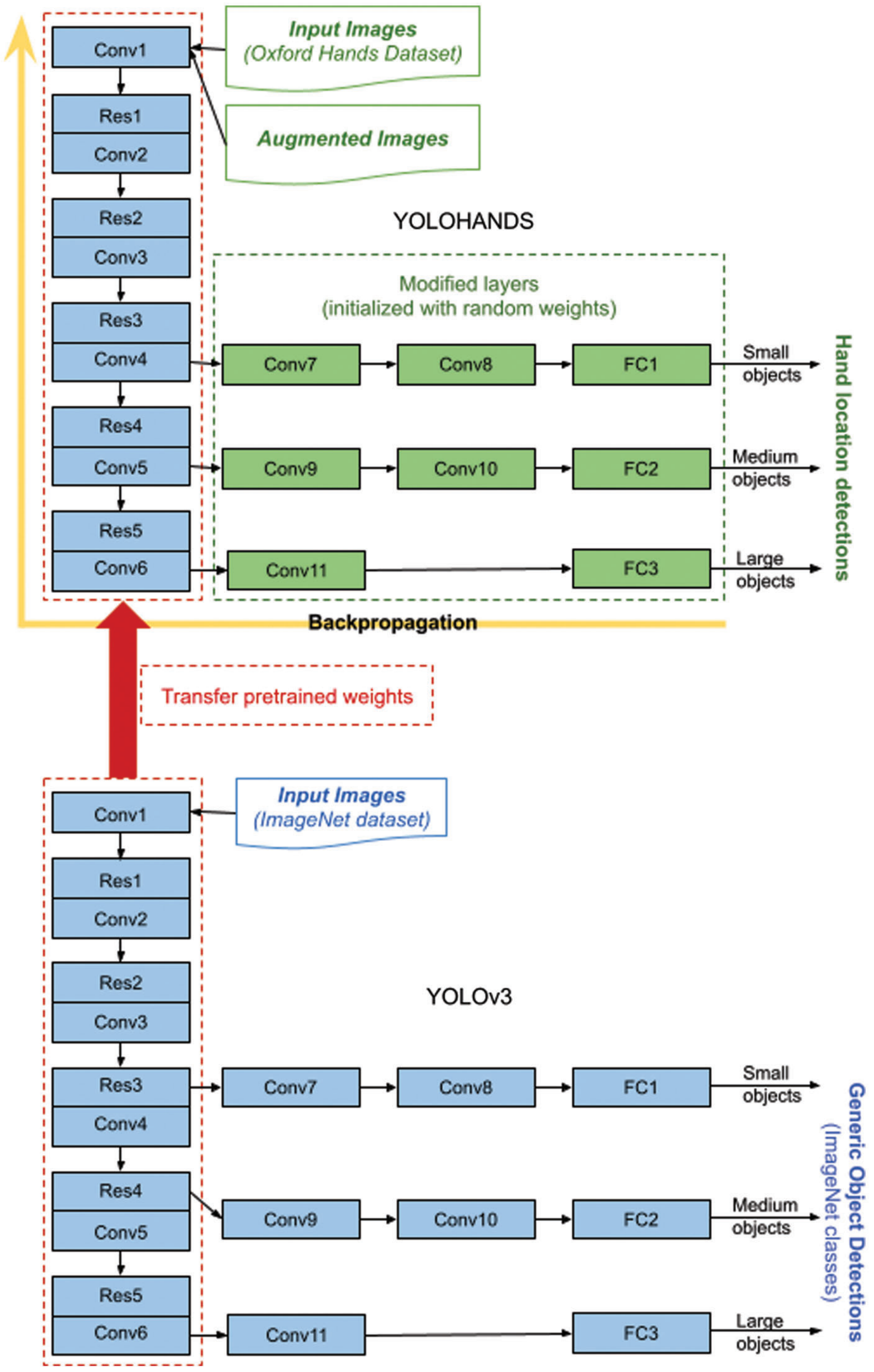

Figure 2: YOLOHANDS deep transfer learning approach based on pretrained YOLOv3 [9]

4.2 Data Augmentation

Importance of data augmentation for visual recognition tasks has been verified through a number of studies [30,31]. We have experimented with different image augmentation techniques and finally formulated an extensive augmentation procedure over the training dataset by generating 38 augmented images for every single image in the training dataset. Before the augmentation, all images are resized to 416 by 416 pixel dimensions. The augmentation starts by rotating the image at an angle that was a random multiple of $\pi/6$. After the rotation, a sequence of transformations has been applied, where each transformation in sequence has a probability of 0.5 to be applied or not. The applied image transformations include: a) addition of random zero mean Gaussian noise; b) gamma adjustment by a gamma factor of 0.4 with gain of 0.9, and c) intensity stretching by truncating the top 2% of lowest and highest intensities. Starting from an initial training set of 9163 hand instances, we formed the augmented training set consisting of $N_{train} = 348194$ instances, which was used to train the YOLOHANDS detector. Note that the augmentation is applied only over a training subset of images, while the test dataset is used in its original form, without any augmentation. Therefore, it will not affect time consumption of the detection inference step. Algorithm of the described augmentation procedure is given in the following:

Algorithm 1: Data augmentation algorithm

```
n_augments = 38;
n_angles = 12;
for i = 1 : n_train_images
   for j = 1 : n_augments
   {
      rand = get_random(0, n_angles-1);
      train_image[i].apply_rotation(rand*PI/6);
      rand_gauss = get_random(0, 3);
      if (rand_gauss > 1)
      {
         train_image[i].apply_gaussian_noise();
      }
      rand_gamma = get_random(0, 3);
      if (rand_gamma > 1)
      {
         train_image[i].apply_gamma_adjust();
      }
      rand_intensity = get_random(0, 3);
      if (rand_intensity > 1)
      {
         train_image[i].apply_ intensity_stretch();
      }
   }
```

5 Evaluation

For quality evaluation of the described YOLOHANDS detection method, we used the following three measures: Recall, Precision and F1 measure. Proper understanding of these measures in the context of object detection domain might be tricky, so we will give a brief description of the corresponding parameters, measures and computation procedures used in the paper. A true positive (TP) is a valid object detection, meaning an object is present in the scene and properly detected. False positive (FP) is an invalid object detection, meaning that an object which is not present in the scene is wrongly detected, or the object is present in the scene, but the detection is not overlapping it. False negative (FN) indicates that an object is present in the scene, but it is not correctly detected. Recall measure calculates the percentage of properly detected objects among all that are present in the dataset, and it can be computed in the following way $Recall = TP/(TP + FN)$. Precision gives the percentage of valid object detections among all detections $Precision = TP/(TP + FP)$. Finally, F1 measure combines Precision and Recall into a single number in the form of a harmonic mean value: $F1 = 2 * (Precision * Recall)/(Precision + Recall)$. Note that all the three measures produce values in the range *[0,1]*, with higher values representing better results. The process of evaluation metrics computation is presented in the following algorithm.

Algorithm 2: Evaluation metrics computation

```
TP = 0; FP = 0; FN = 0;
for i = 1 : n_annotations
{
    IoU = 0
    for j = 1 : n_detections
    {
        IoU_current = IOU(annotation[i], detection[j]);
        IoU = max(IoU, IoU_current);
    }
    if (IoU > 0.25)
    {
        TP += 1;
    }
}
FN = n_annotations-TP;
FP = n_detections-TP;
```

To measure if a single detected bounding box is correctly positioned relative to the annotation, an intersection over union (*IoU*) is computed between detected and annotated bounding boxes. *IoU* takes values from the interval [0,1], with value 0 meaning there is no intersection at all, and value 1 meaning that the detected and annotated boxes perfectly overlap. During testing, for every annotation we compute *IoU* with each of the detected bounding boxes and then take the maximum value. If, the value is above the threshold of 0.25, we will count that detection as valid and increment the TP score. To compute the FN score, we simply subtract the TP score from the number of annotations in an image. To compute the FP score, we subtract the TP score from the number of detected objects.

Training of the network is performed in batches of $N_{batch} = 64$ images. Note that the YOLO framework reports the number of training iterations instead of more common number of training epochs, but it can be easily converted as: $N_{epoch} = N_{iter} * N_{batch}/N_{train}$. The quantitative results of the evaluation on the training subset of the Oxford Hands Dataset, depending on the number of training iterations, are presented in Tab. 1.

Table 1: Quantitative results of the YOLOHANDS on the training subset

Num. train iterations (in thousands)	Precision	Recall	F1
10	0.83	0.62	0.71
11	0.82	0.70	0.75
12	0.83	0.67	0.74
13	0.81	0.72	0.76
14	0.82	0.72	0.77
15	0.83	0.70	0.76
16	0.80	0.74	0.77
17	0.84	0.72	0.78
18	0.81	0.73	0.77
19	0.83	0.72	0.77
20	0.82	0.73	0.77
21	0.82	0.75	0.78
22	0.80	0.72	0.75
23	0.83	0.73	0.78
24	0.80	0.73	0.77
25	0.83	0.72	0.77

From the results given in Tab. 1, it can be noticed that Precision has a relatively stable value among all training checkpoints. On the other hand, the Recall and F1 measures show an increase in value until the 16000 training iterations (~3 training epochs), while stabilizing afterwards. If F1 score is assumed as a single measure of quality, it can be noticed that the optimal result of F1 = 0.78 is given after 21 k iterations (~4 training epochs). Therefore, for further experiments we will use the weights after 21 k iterations, where P = 0.82 and R = 0.75. Some qualitative results from the YOLOHANDS method can be seen in Fig. 4.

A commonly used single quality measure for evaluation of object detectors is Average Precision (AP), which is calculated as the area under the precision-recall curve. To generate the curve, we used the YOLOHANDS model weights with the optimal F1 score, and varied detection confidence thresholds (as described in 3.2). The computed precision and recall results are given in Tab. 2, with the corresponding curve presented in Fig. 3 below.

We further compared results of our approach with state-of-the-art results from the literature, evaluated on the same Oxford Hands Dataset (see Tab. 3).

Table 2: Precision and recall measures computed by varying detection confidence

Detection confidence	1.0	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1	0.0
Precision	1.0	0.99	0.97	0.96	0.92	0.89	0.87	0.84	0.80	0.71	0.26
Recall	0.0	0.10	0.28	0.42	0.52	0.61	0.68	0.73	0.77	0.81	0.90

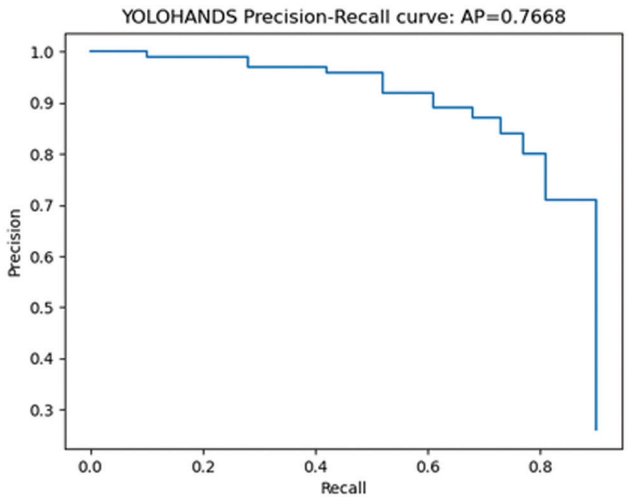

Figure 3: Precision-recall curve used to compute the final average precision AP = 76.68%

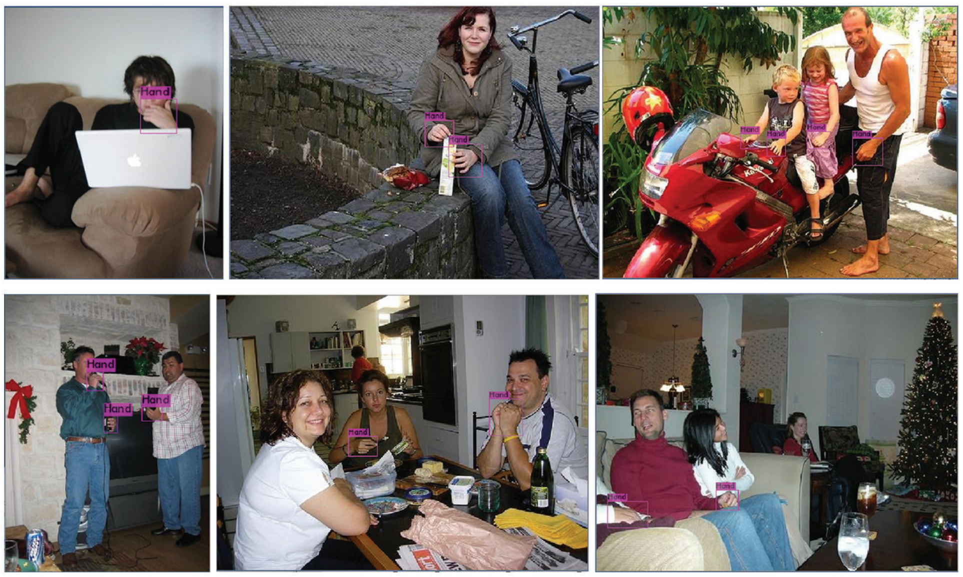

Figure 4: Examples of detection results of the YOLOHANDS method (detected hands are marked with pink bounding boxes)

The obtained value of AP = 76.68% represents the top result on this dataset. This confirms the adaptation power of generic YOLO-based neural network architectures and its ability to successfully handle specific detection tasks, such as human hand detection. Some qualitative results of our YOLOHANDS detection approach in challenging environments, are given in Fig. 4.

Table 3: Comparison of our method with state-of-the-art methods on the oxford hands dataset

Method	Average precision (%)
YOLOHANDS (ours)	76.68
MS-RFC [2]	75.10
Mittal et al. [14]	48.20
Mittal et al. (baseline) [14]	33.60

6 Conclusion

Robust and accurate detection of human hands in uncontrolled environments is a challenging visual recognition task, which is of crucial importance for many high-level human behavior analysis tasks. In this paper we proposed and evaluated a method for robust hand detection which is built by a deep transfer learning approach over YOLO neural network architecture. The initial model is adapted to the single class hand detection task, by modifying the higher convolutional layers including the last fully connected layer, while initializing lower non-modified layers with the generic pretrained weights. To address robustness issues of the overall method, we proposed a comprehensive data augmentation procedure over the training image dataset. The obtained experimental results on a challenging publicly available dataset show the highest level of accuracy of the proposed method, which is comparable to the state-of-the-art methods. It can be concluded that state-of-the-art generic object detection methods, such as YOLO, can be successfully adapted to the specific hand detection task by deep transfer learning approach and adequate data preprocessing procedure. The presented results encourage experimenting with other highly accurate object detection methods, such as Faster-RCNN [7] or YOLOS Transformer [21].

Funding Statement: The research presented in this paper is financed by the Ministry of Education, Science and Technological Development of the Republic of Serbia.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] K. M. Sagayam and D. J. Hemanth, "Hand posture and gesture recognition techniques for virtual reality applications: A survey," *Virtual Reality*, vol. 21, no. 2, pp. 91–107, 2017.

[2] T. H. N. Le, K. G. Quach, C. Zhu, C. N. Duong, K. Luu *et al.,* "Robust hand detection and classification in vehicles and in the wild," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition Workshops*, Honolulu, USA, pp. 1203–1210, 2017.

[3] D. Bragg, O. Koller, M. Bellard, L. Berke, P. Boudreault *et al.,* "Sign language recognition, generation, and translation: An interdisciplinary perspective," in *Proc. of the 21st Int. ACM SIGACCESS Conf. on Computers and Accessibility*, Pittsburg, USA, pp. 16–31, 2019.

[4] J. Gangrade, J. Bharti and A. Mulye, "Recognition of Indian sign language using ORB with bag of visual words by kinect sensor," *IETE Journal of Research*, vol. 69, no. 1, pp. 1–15, 2020.

[5] R. Girshick, "Fast R-CNN," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, Boston, USA, pp. 1440–1448, 2015.

[6] R. Girshick, J. Donahue, T. Darrell and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, Columbus, USA, pp. 580–587, 2014.

[7] S. Ren, K. He, R. Girshick and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. of the Advances in Neural Information Processing Systems*, Montreal, Canada, pp. 91–99, 2015.

[8] J. Redmon and A. Farhadi, “YOLO9000: Better, faster, stronger,” in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, Honolulu, USA, pp. 7263–7271, 2017.

[9] J. Redmon and A. Farhadi, “YOLOv3: An incremental improvement,” *arXiv preprint*, arXiv:1804.02767, 2018.

[10] J. Yosinski, J. Clune, Y. Bengio and H. Lipson, “How transferable are features in deep neural networks?,” in *Proc. of the Advances in Neural Information Processing Systems*, Montreal, Canada, pp. 3320–3328, 2014.

[11] A. S. Razavian, H. Azizpour, J. Sullivan and S. Carlsson, “CNN features off-the-shelf: An astounding baseline for recognition,” in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition Workshops*, Columbus, USA, pp. 806–813, 2014.

[12] N. S. A. Hanan, A. Hosni Mahmoud and A. H. Alharbi, “Time-efficient fire detection convolutional neural network coupled with transfer learning,” *Intelligent Automation & Soft Computing*, vol. 31, no. 3, pp. 1393–1403, 2022.

[13] R. J. K. R. P. Narmadha and N. Sengottaiyan, “Deep transfer learning based rice plant disease detection model,” *Intelligent Automation & Soft Computing*, vol. 31, no. 2, pp. 1257–1271, 2022.

[14] A. Mittal, A. Zisserman and P. H. Torr, “Hand detection using multiple proposals,” in *Proc. of the British Machine Vision Conf.*, Dundee, UK, pp. 1–11, 2011.

[15] Z. G. Darehnaei, S. M. J. R. Fatemi, S. M. Mirhassani and M. Fouladian, “Ensemble deep learning using faster R-CNN and genetic algorithm for vehicle detection in UAV images,” *IETE Journal of Research*, vol. 70, no. 1, pp. 1–10, 2021.

[16] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed *et al.,* “SSD: Single shot multibox detector,” in *Proc. of the European Conf. on Computer Vision*, Amsterdam, Netherlands, pp. 21–37, 2016.

[17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones *et al.,* “Attention is all you need,” in *Proc. of the Advances in Neural Information Processing Systems*, Long Beach, USA, pp. 5998–6008, 2017.

[18] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai *et al.,* “An image is worth 16 × 16 words: Transformers for image recognition at scale,” in *Proc. of the Int. Conf. on Learning Representations*, Vienna, Austria, pp. 1–21, 2021.

[19] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei *et al.,* “Swin transformer: Hierarchical vision transformer using shifted windows,” in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 10012–10022, 2021.

[20] T. Chen, S. Saxena, L. Li, D. J. Fleet and G. Hinton, “Pix2seq: A language modeling framework for object detection,” in *Proc. of the Int. Conf. on Learning Representations*, pp. 1–17, 2022.

[21] Y. Fang, B. Liao, X. Wang, J. Fang, J. Qi *et al.,* “You only look at one sequence: Rethinking transformer in vision through object detection,” in *Proc. of the Advances in Neural Information Processing Systems*, New Orleans, USA, pp. 1–18, 2021.

[22] M. Oquab, L. Bottou, I. Laptev and J. Sivic, “Learning and transferring mid-level image representations using convolutional neural networks,” in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition Workshops*, Columbus, USA, pp. 1717–1724, 2014.

[23] M. A. R. Khan and M. K. Jain, “Feature point detection for repacked android apps,” *Intelligent Automation & Soft Computing*, vol. 26, no. 6, pp. 1359–1373, 2020.

[24] H. Sun and R. Grishman, “Lexicalized dependency paths based supervised learning for relation extraction,” *Computer Systems Science and Engineering*, vol. 43, no. 3, pp. 861–870, 2022.

[25] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, “You only look once: Unified, real-time object detection,” in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition Workshops*, Las Vegas, USA, pp. 779–788, 2016.

[26] T. -Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona *et al.,* “Microsoft COCO: Common objects in context,” in *Proc. of the European Conf. on Computer Vision*, Zurich, Switzerland, pp. 740–755, 2014.

[27] A. Bochkovskiy, C. -Y. Wang and H. -Y. M. Liao, “YOLOv4: Optimal speed and accuracy of object detection,” *arXiv preprint*, arXiv:2004.10934, 2020.

[28] K. He, X. Zhang, S. Ren and J. Sun, “Deep residual learning for image recognition,” in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition Workshops*, Las Vegas, USA, pp. 770–778, 2016.

[29] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh *et al.,* "ImageNet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[30] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, no. 1, pp. 60, 2019.

[31] A. Robey, G. J. Pappas and H. Hassani, "Model-based domain generalization," *Advances in Neural Information Processing Systems*, vol. 34, no. 1, pp. 20210–20229, 2021.