Scientific Visualization, 2024, volume 16, number 2, pages 154 - 168, DOI: 10.26583/sv.16.2.11

Dual-Pass Feature-Fused SSD Model for Detecting Multi-Scale Vehicles on the Construction Site

M. Petrov[1,A], S. Zimina[2,A], D. Dyachenko[3,B], A. Dubodelov[4,B], S. Simakov[5,A]

[A] Moscow Institute of Physics & Technology,
Instituskii Lane 7, Dolgoprudny 141700, Russia
[B] LLC Acceleration, Stolyarnii Lane 3, Moscow 115114, Russia

[1] ORCID: 0000-0003-4907-4687, mikhail.petrov@phystech.edu
[2] ORCID: 0000-0003-4160-9915, sofya.zimina@phystech.edu
[3] ORCID: 0009-0004-5158-9377, d.dyachenko@acceleration.ru
[4] ORCID: 0009-0000-9162-5863, a.dubodelov@acceleration.ru
[5] ORCID: 0000-0003-3406-9623, simakov.ss@phystech.edu

Abstract

When detecting equipment on a construction site the objects of detection could have very different scale relative to the image on which they are located. For better detection and bounding box visualization of small objects, a Feature-Fused modification of the SSD detector can be used. Together with the use of overlapping image slicing on the inference, this model copes well with the detection of small objects. However, excessive manual adjustment of the slicing parameters for better detection of small objects can both generally worsen detection on scenes different from those on which the model was adjusted, and lead to significant losses in the detection of large objects and problems with their bounding box visualization. Therefore, to achieve the best quality, the image slicing parameters should be automatically selected by the model depending on the characteristic scales of objects in the image.

The article presents a dual-pass version of Feature-Fused SSD for automatic determination of image slicing parameters. To determine the characteristic sizes of detected objects on the first pass, a fast truncated version of the detector is used. On the second pass the final object detection is carried out with slicing parameters selected after the first one. Depending on the complexity of the task being solved, the detector demonstrates a quality of 0.82 - 0.92 according to the mAP (mean Average Precision) metric.

Keywords: computer vision; construction site; construction vehicles; single shot detector.

1. Introduction

Detection and bounding box visualization of construction equipment at a construction site is a natural requirement for a wide class of construction control tasks. Examples of such tasks include monitoring progress, tracking work, and labor protection. The construction site can be monitored using surveillance cameras. In order to form a complete picture of the work being carried out, such cameras are installed to cover as large an area of the construction site as possible. In this case, the image from the surveillance camera may contain many objects of different (mostly small) scale relative to the size of the image itself. Such objects are often poorly distinguishable even by the human eye. Automation of detection of objects from surveillance cameras significantly speeds up the process of image analysis and objects bounding box visualization.

In connection with the development of deep machine learning methods, neural network approaches have become widespread, which allow solving a wide class of computer vision

problems and, in particular, object detection [5]. There are two key types of architectures for object detection. The first type is onestage approaches such as SSD [6], YOLO [7], RFBNet [8]. The second type is two-stage approaches such as Faster-RCNN [9], Mask-RCNN [10], ReasoningRCNN [11]. In two-stage approaches, the model first proposes a set of regions of interest using, for example, a selective search. The classifier then processes only candidates from this set. The one-stage approach skips the ROI suggestion stage and runs discovery directly on a dense set of possible locations that is defined by the neural network architecture. Therefore, one-stage approaches are usually more computationally fast and are more widely used in practice. Among such architectures, SSD and YOLO can be distinguished.

The use of neural network approaches is also widespread for solving computer vision problems at a construction site. In work [4] the IFaster R-CNN model is used to detect workers and construction equipment on site. The work demonstrates the high accuracy of the presented model and some successes for detection, including small objects. In the article [2] to detect equipment, it is proposed to use the MobileNet SSD detector. Judging by the examples presented in the work, the model is intended for the detection of large objects. Authors of the work [3] propose an R-FCN model built using transfer learning for equipment detection. In work [4] the Faster R-CNN model is used for the detection of workers and construction equipment.

The basic version of the SSD detector compresses the input image to a size 300 × 300 pixels. Therefore, if the input image has a higher resolution, and the objects of interest in the image are small, then such an object will most likely not be found. To solve the problem of detecting small objects, you can compress the input image less - use the SSD, which compresses the input image to the size 512 × 512. As a next step, you can use the Feature Fusion SSD [12]. This approach allows you to deal with many problems of the basic implementation, in particular, in addition to better finding small objects, combine features of different scales. This is achieved by combining features from different layers of the network with different scales and thereby creating a new feature map. If this is not enough to detect objects, you can use image slicing with overlaps [13] as a post processor. This approach allows us to consider one image as a collection of its parts. It is clear that small objects for each part of the original image will have a larger relative size than they had in the original image. But in this case, you may encounter the following problem: the slicing parameters selected for one task may not be optimal for another. Also, the choice of slicing parameters, based only on the best detection of small objects, may adversely affect the detection of larger objects. To automatically determine the image slicing parameters, a two-pass detector, proposed in this article, can be used, in which, on the first pass, a fast truncated version of the detector (SSD) is used, which allows determining the characteristic sizes of objects to be detected, and on the second pass, the final detection of objects with parameters cuts selected after the first pass. The result of the study shows that the proposed model makes it possible to detect objects at a construction site with high accuracy. The proposed slicing approach consists of data pre-processing and model predictions post-processing and can be used with any detector model. We compared the performance of two most popular one-shot detectors: SSD and YOLO on segment from our dataset. We also tested FSSD model (SSD architecture, modified for better small object detection) on this data. FSSD512 showed better results than two other models (in terms of accuracy on scenes with small and various objects), so we chose it as detector model for our algorithm.

The article has the following structure: in the section 2 we present our dataset. In section 3 the main neural network architecture (3.1), the slicing algorithm (3.2) and the main idea of constructing a two-pass model (3.3) are described. The results of applying the proposed model are presented in the section 4. The conclusion is given in the section 5.

2. Dataset

For training and testing the model, a hand-crafted dataset of construction equipment was used. The dataset consists of 1450 photos of equipment at a construction site (1200 — training set, 140, — validation set, 110 — test set). There are 7 different classes of construction vehicles in the photos: excavator, truck crane, front loader, roller, bulldozer, dump truck, truck. Objects have different scales relative to the image (in the training sample there are ≈ 600 photos with large and medium objects, ≈ 300 with small objects and ≈ 300 with diverse objects; in the test sample there are ≈ 60 images with large and medium objects, ≈ 25 — with small ones and ≈ 25 — with diverse ones). Examples of images from the dataset are shown in the figure. 1.

Figure 1. Examples of images from the dataset

In order to improve model performance in small object detection we applied slicing approach described in 3.2 to train images and added obtained patches to train data.

3. Methods

3.1 CNN architecture

3.1.1 SSD

In this work we use a neural network based on SSD512 (Single Shot Detector) architecture. The input size of this model is 512 × 512 pixels. Figure 2 shows the architecture details of the model. VGG-16 network and additional convolution layers are used for feature extraction. The additional layers decrease in size progressively and allow predictions of detections at multiple scales. VGG-16 top layers (*conv4_3, conv7*) and additional layer outputs are used as feature maps. The feature maps have sizes of 64 × 64, 32 × 32, 16 × 16, 8 × 8, 4 × 4, 2 × 2 and 1 × 1. Each feature map cell is associated with a set of default bounding boxes (prior boxes), that are located in the center of the cell and vary over aspect ratio (1 × 1, 1 × 2, 2 × 1, 1 × 3, 3 × 1 and an additional square box of larger size). Each feature map is fed into a corresponding output layer that predicts the offsets relative to the prior box shapes in the cell, as well as the per-class scores. For example, for the ouput of the layer *conv4_3* with size 64 × 64 the model predicts 64 × 64 × 4 = 16384 boxes (feature map of this layer has 4 prior boxes per cell).

SSD512 model predicts shape offsets and class scores for 24564 prior boxes (64 × 64 × 4 + 32 × 32 × 6 + 16 × 16 × 6 + 8 × 8 × 6 + 4 × 4 × 6 + 2 × 2 × 4 + 1 × 1 × 4 = 24564).

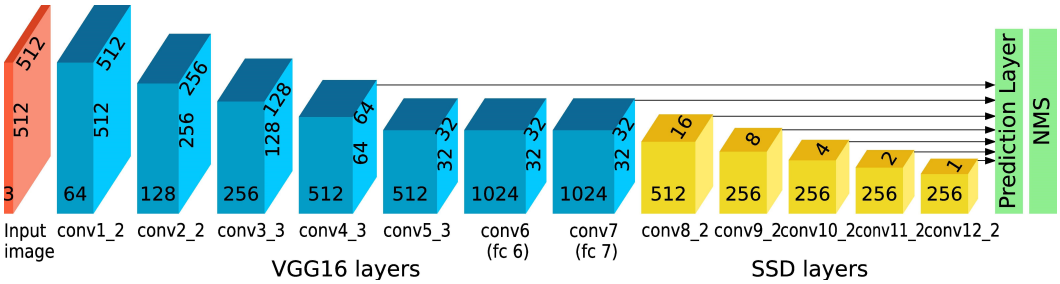


Figure 2. SSD512 architecture

3.1.2 FSSD

In order to improve accuracy for small objects we use FSSD (Feature-Fused SSD) model, illustrated in figure 3. Instead of using the feature map of *conv4_3* layer, FSSD uses output from a feature-fusion module that combines feature maps of *conv4_3* and *conv4_3* layers. The feature-fusion module is shown in figure 4. In this work we use a concatenation feature-fusion module, where *conv4_3* and *conv4_3* layers outputs are concatenated along their channel axis. In order to make the feature maps of conv5_3 layer the same size as conv4_3 layer, the conv5_3 layer is followed by a deconvolution layer, which is initialized by bilinear upsample. Before concatenation, the feature maps are fed to normalization layers with different scales respectively, e.g. 10, 20. The final feature fusion maps are generated by a 1 × 1 convolutional layer for dimension reduction as well as feature recombination. SSD uses shallow layers to predict smaller objects; because of that using feature fusion allows to improve the detection performance of small objects.

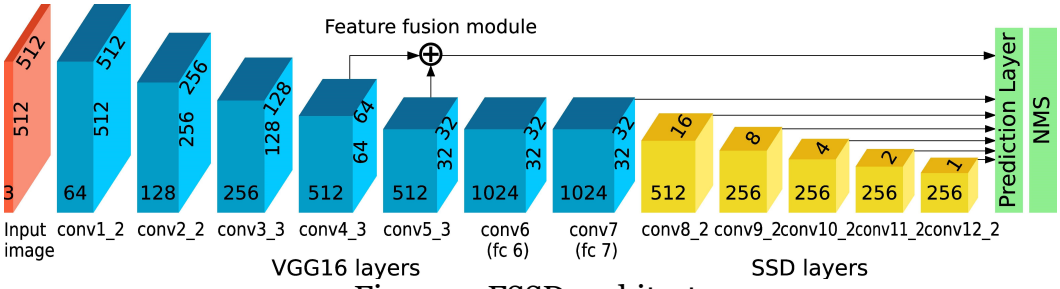


Figure 3. FSSD architecture

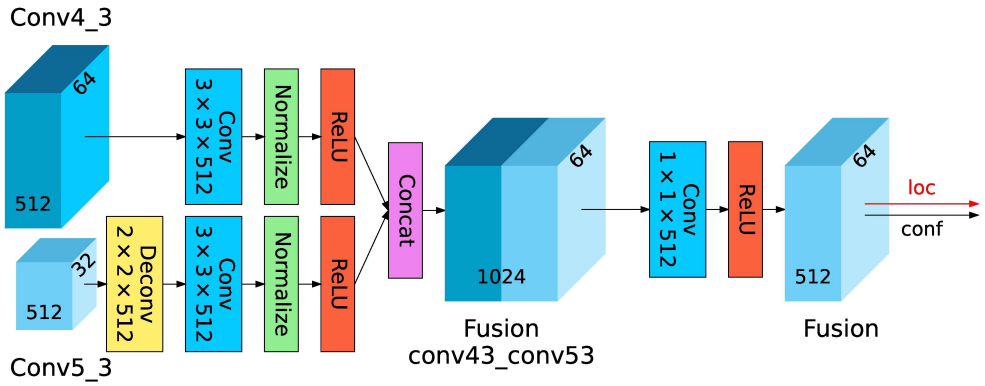


Figure 4. Feature-Fusion

3.1.3 NMS (Non-Maximum Suppression)

The SSD model uses several feature maps and predicts multiple bounding boxes for each cell of the feature map, therefore several bounding boxes could correspond to the same ground truth bounding box. In order to select the most relevant predictions, Non-Maximum Suppression algorithm is used:

- predicted boxes are sorted in descending order by their corresponding scores
- pairwise IoU (Intersection over union) scores are computed
- if *IoU* value between two boxes is above the threshold (*IoU* > *max_overlap*), these boxes will be considered to correspond the same object, and the box with lower score will be suppressed.

An example of predicted boxes before and after applying NMS is shown in figure 5.

(a) Before NMS (b) After NMS

3.1.4 Loss function

We used the MultiBox loss function described in [6]. Let $x_{ij}^{p} = 1,0$ be an indicator for matching the *i*-th default box to the *j*-th ground truth box of category *p*. The overall objective loss function is a weighted sum of the localization loss (loc) and the confidence loss (conf):

$$L(x,c,l,g) = \frac{1}{N}(L_{conf}(x,c) + \alpha * L_{loc}(x,l,g)),$$

Figure 5. Predicted boxes before (a) and after (b) applying NMS

where *N* is the number of matched default boxes. If *N* = 0, wet set the loss to 0.

The localization loss is a Smooth L_1 loss between the predicted box (*l*) and the ground truth box (*g*) parameters:

$$L_{loc}(x,l,g) = \sum_{i \in Pos}^{N} \sum_{m \in cx,cy,w,h} x_{ij}^{k} smoothL_1(l_i^m - \hat{g}_j^m),$$

$$\hat{g}_j^{cx} = \frac{g_j^{cx} - d_i^{cx}}{d_i^w}, \hat{g}_j^{cy} = \frac{g_j^{cy} - d_i^{cy}}{d_i^h},$$

$$\hat{g}_j^w = \log\left(\frac{g_j^w}{d_i^w}\right), \hat{g}_j^h = \log\left(\frac{g_j^h}{d_i^h}\right),$$

where *d* is the default (prior) bounding box, *cx, cy* are the coordinates of box center, *w, h* are the box width and height. The confidence loss is the softmax loss over multiple classes confidences (*c*). We set the α coefficient to 1.

3.2 Slicing algorithm

In order to improve the detection performance of small objects we use a slicing algorithm. The relative object sizes are increased in the cropped image compared to the initial input image. In this work, we apply slicing both to train and test data.

3.2.1 Slicing on train

Our algorithm splits each image from train data into $n \times m$ equal overlapping patches. Ground truth bounding boxes list for each patch contains only boxes whose centers are inside this patch. We use the following parameters in our algorithm:

(1) *tiles_n, tiles_m* — the number of horizontal and vertical splits respectively.

(2) *inter_w, inter_h* — the percentage overlap between neighboring patches

Figure 6 shows an example of a split image, borders of each patch are highlighted with colored lines.

Figure 6. Example of a split image

3.2.2 Slicing on test

The input test image is split into overlapping patches, then all the patches (together with initial image) are fed into the neural network and finally the detection results obtained for each patch are merged. In order to merge the predictions and filter out irrelevant detections (the model predicts multiple boxes, most of them have low confidence score and several boxes cound correspond to the same ground truth object) we use a postprocessing algorithm as follows:

(1) convert detected boxes coordinates to absolute coordinates in initial image

(2) filter predicted boxes by confidence score (*score* < *min_score*)

(3) perform local NMS (*overlap* > *max_overlap* for each patch and each class separately).

(4) perform global NMS (*overlap* > *max_overlap_global* for entire image but for each class separately)

(5) perform global multiclass NMS (*overlap* > *max_overlap_multiclass* for the entire image and all classes). This step is important in case of multiclass detection, when the objects belonging to different classes could overlap. When we detect vehicles on the construction site, this can be, for example the case when an excavator is loading sand into a dump truck. In order to differ such cases from cases of multiple detections for one object, we use multiclass NMS.

Different thresholds may be used for different NMS steps as well as for different classes.

3.3 Two-pass detection

The proposed algorithm has some disadvantages. Since the slicing parameters are preselected for all images, the method is only applicable to images of similar scale. Oversized patches result in low quality predictions for small objects, undersized patches lead to multiple detections for the same object (figure 7).

Figure 7. Undersized patches could lead to multiple detections for the same object

In order to avoid multiple detections, a 2-pass algorithm can be used. The first pass is used to obtain the optimal number of slices. During the first pass, the input image is divided into fixed default number of patches. Then the patches and the initial image are fed into the detector network, and the obtained boxes are filtered by confidence score (*score* > *min_score_prev*). For computational efficiency, we don't apply NMS during the first pass. Filtered boxes are used to compute the mean box size, and based on this size the optimal number of slices *tiles_n*, *tiles_m* is computed (the optimal size of patch is: *patch_size* = *mean_size*·*rec_coeff*). For our data we use *rec_coeff* values within the range of 6 – 8. During the second pass the image is divided into the optimal number of slices obtained from the first pass. Then the same detection algorithm is used as for 1 pass detection.

3.4 Metric

In our work we use two metrics to compare the quality of different methods: *Accuracy* and *Mean Average Precision (mAP)*.

3.4.1 Accuracy

Accuracy metric is calculated as follows:

$$Accuracy = \frac{TP}{TP + FP + FN}$$

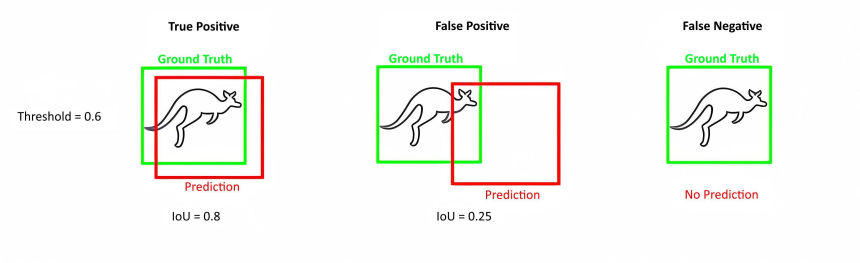


Figure 8. True positive, false positive, false negative

Here, true positive (*TP*) means the match of the predicted box with the real box of the object (*IoU* between the predicted and real box above the threshold), false positive detection (false positive, *FP*) means the absence of matches of the predicted box with all real boxes of objects of this class (*IoU* below the threshold), false negative (*FN*) — absence of a predicted box corresponding to a real box (figure 8). These values are calculated for each class separately, and when calculating the accuracy metric, their total values for all classes are used.

3.4.2 mAP

mAP metric (*Mean Average Precision*) is calculated using the method described in [14]. First step is to determine if each predicted box is true positive or false positive. Then boxes are sorted in descending order by score, and then cumulative values of TP and FP are calculated as follows:

$$TP_i^{cumulative} = \sum_{j=0}^{i} TP_j$$

$$FP_i^{cumulative} = \sum_{j=0}^{i} FP_j$$

Based on obtained cumulative *TP* and *FP* values cumulative *precision* and *recall* are computed:

$$precision = \frac{TP}{TP + FP} = \frac{TP}{n_{predictions}}$$

$$recall = \frac{TP}{TP + FN} = \frac{TP}{n_{objects}}$$

Average Precision (*AP*) is calculated as the mean precision at a set of eleven equally spaced recall levels [0, 0.1, ..., 1]. The precision at each recall level is interpolated by taking the maximum precision measured for all cumulative precisions for which the corresponding recall exceeds the threshold.

$$AP = \frac{1}{11} \sum_{recall \in \{0, 0.1, \ldots, 1\}} \max_{\tilde{r} \geq recall} (precision(\tilde{r}))$$

Both Accuracy and mAP metrics were calculated with *IoU* threshold of 0.5.

4. Results

We trained out model for 125 epochs. 9 shows the model loss on train and validation data.

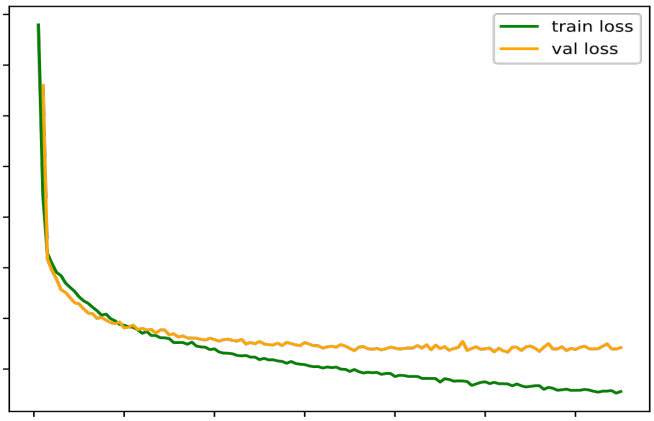
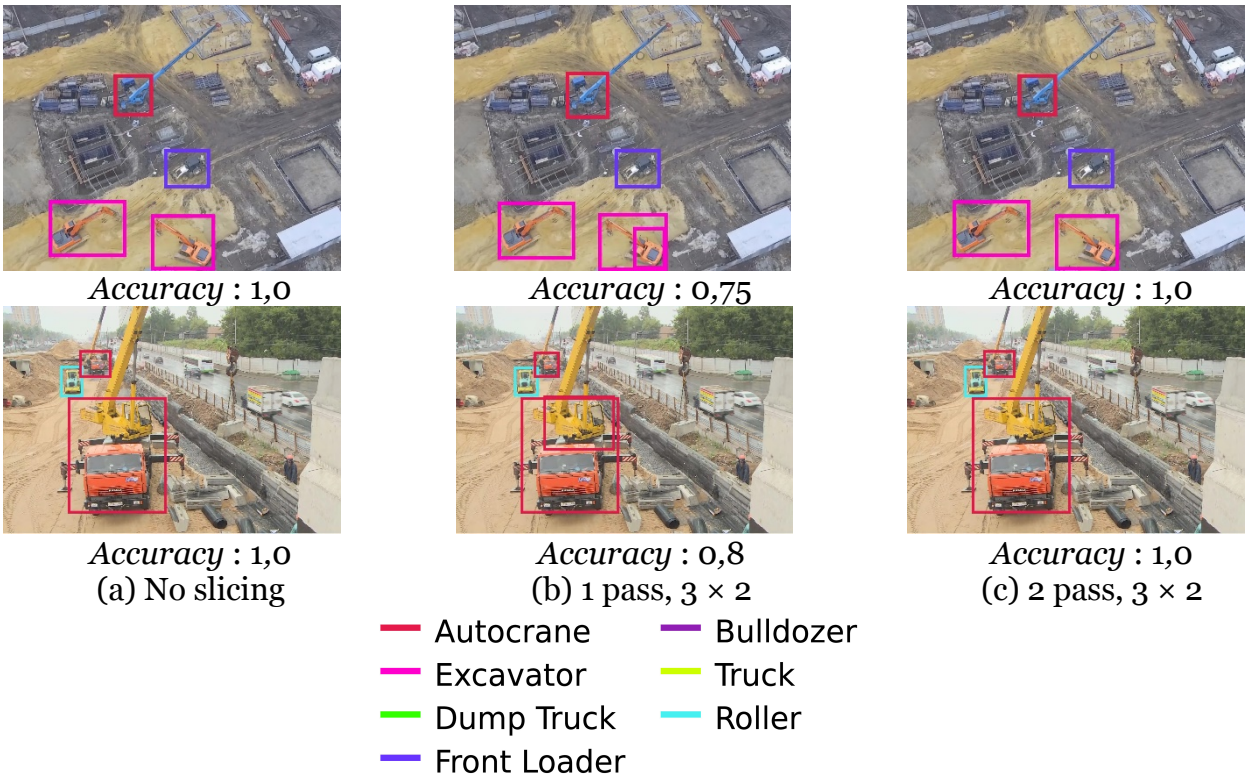


Figure 9. Train and validation loss

4.1 Methods comparison

The results of two-pass detection (with determination of optimal slicing parameters) were compared with one-pass detection (fixed slicing parameters) and detection without partitioning. The FSSD512 architecture was used as a detector in all three cases. Below are the results for each of the methods at different sizes of objects relative to the image (large, small and diverse). Also, for each detection result, the Accuracy metric was calculated.

The figure 10 shows the results of detection of relatively large objects in the image for the FSSD model without slicing, with slicing and one pass of the network, and with optimal slicing of the two-pass model. It can be seen from the figure that the base model, like the two-pass model, copes with the task equally well. On the other hand, the use of slicing for the detection of large objects reduces the quality of prediction. This example justifies the use of a two-pass model to select the optimal slicing parameters.



(a) No slicing (b) 1 pass, 3 × 2 (c) 2 pass, 3 × 2

Figure 10. Detection results for (a) algorithm without slicing, (b) with 3×2 slicing, and (c) two-pass algorithm with initial slicing 3×2. Vertically - the main image and some enlarged parts of it (dotted on the main image).

The figure 11 shows examples of using the FSSD model without slicing, with slicing and one pass of the network, and with the optimal slicing of the twopass model for the detection of very small objects. The figure in the first row shows the main image and for each of the images vertically some of the most significant parts of it are presented. It can be seen that both the base model and the slicing model give low quality predictions. The two-pass model makes it possible to significantly increase the quality of prediction of small objects due to the automatic selection of image slicing parameters.

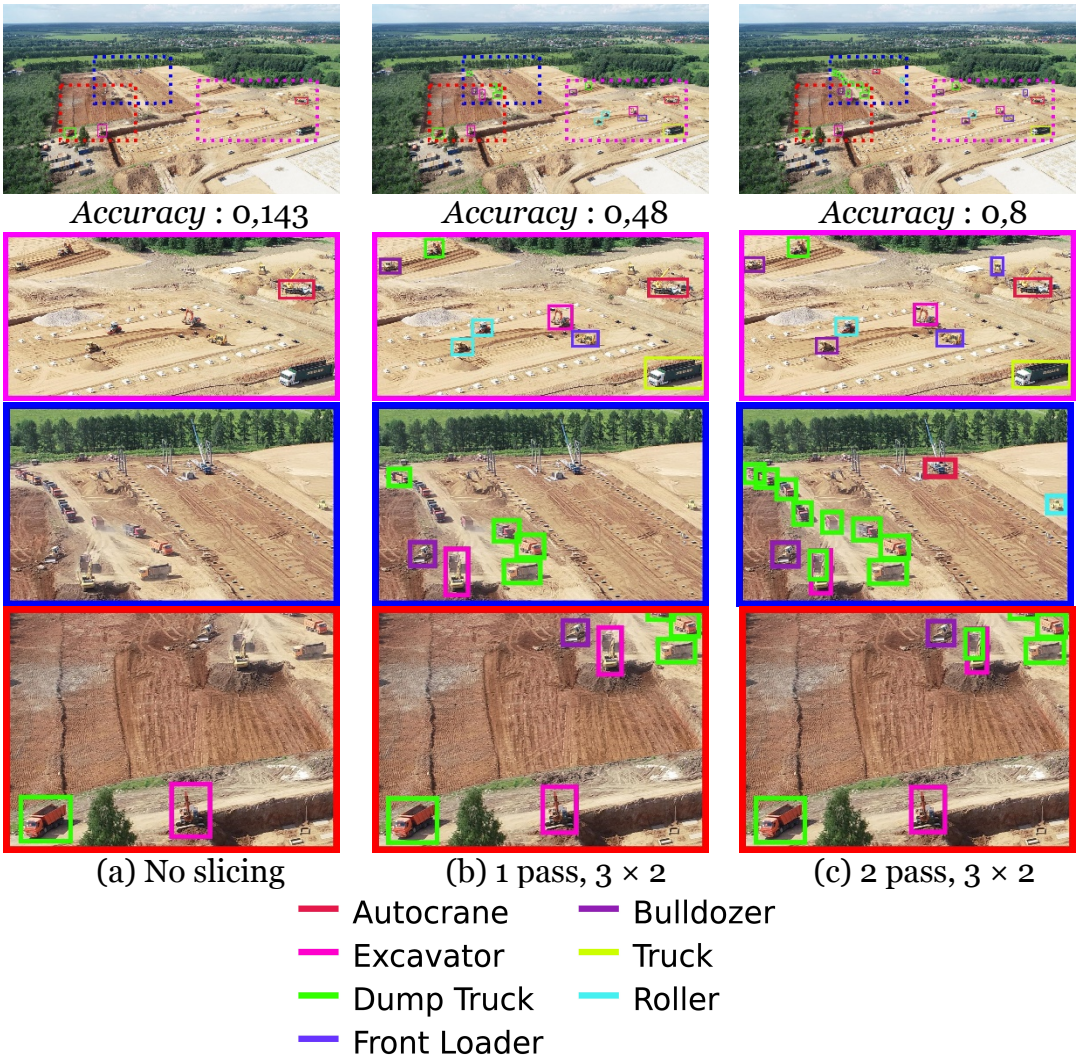


Figure 11. Detection results for (a) algorithm without slicing, (b) with 3×2 slicing, and (c) two-pass algorithm with initial slicing 3×2. Vertically - the main image and some enlarged parts of it (dotted on the main image).

Figure 12 demonstrates the best predictive power of the two-pass model for detecting background objects. This is especially evident in the examples presented in the second row.

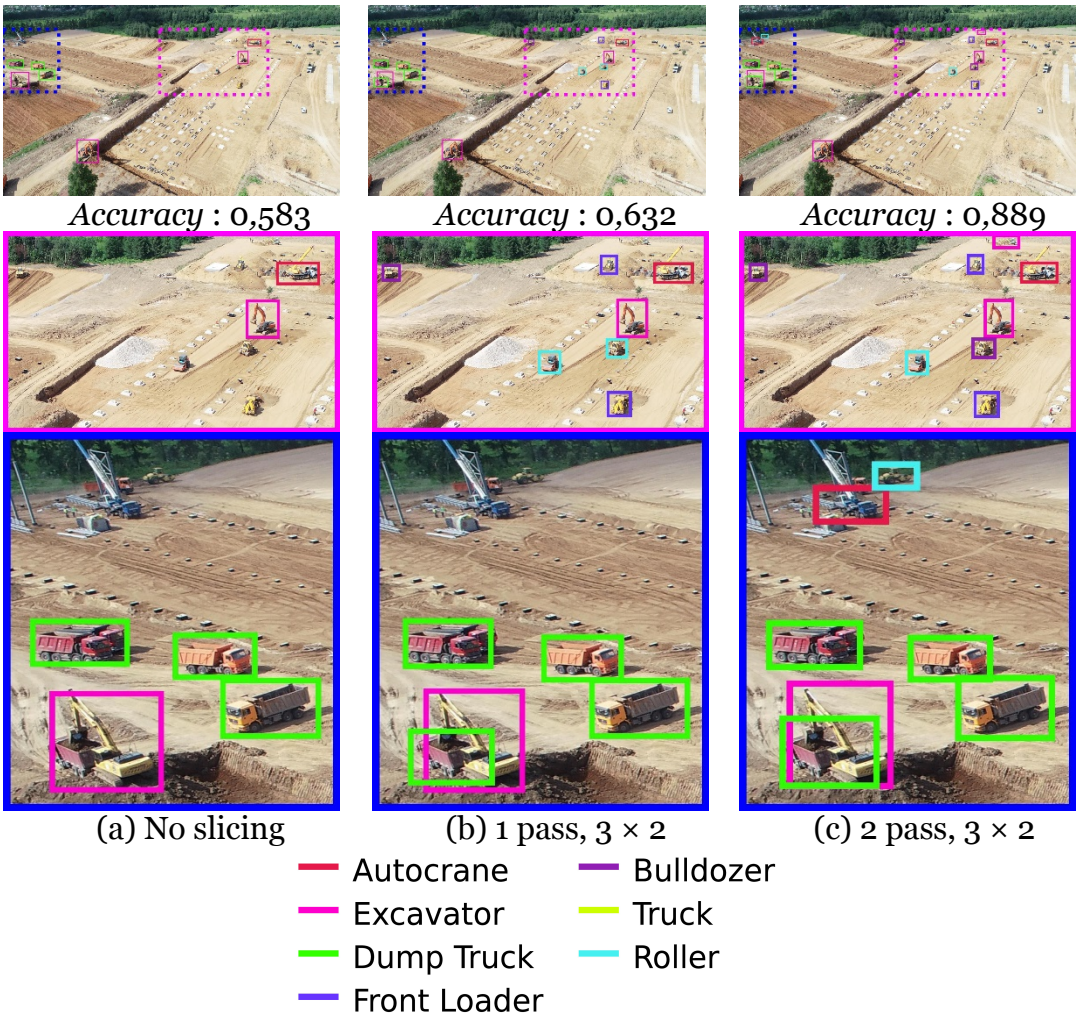


Figure 12. Detection results for (a) algorithm without slicing, (b) with 3×2 slicing, and (c) two-pass algorithm with initial slicing 3×2. Vertically - the main image and some enlarged parts of it (dotted on the main image).

The final example in this series is the 13 figure. This example is characterized by a high density of objects in both the foreground and background. As can

be seen, again the prediction of the background objects is better in the case of using a two-pass model.

The figure 14 shows confusion matrices for each method. Each cell consists the number of objects from class *A* (column headers) that were predicted as class *B* (row headers) by model. In bracket the relative number in percent is shown (divided by number of ground truth objects of this class and multiplied by 100).

The 1 table contains general statistics on the *mAP* metric for each method on the entire dataset and for each part of it separately, characterized by the size of objects (large objects, small objects and diverse objects). The table shows that the two-pass detector gives the best prediction both in general and for each subgroup separately.

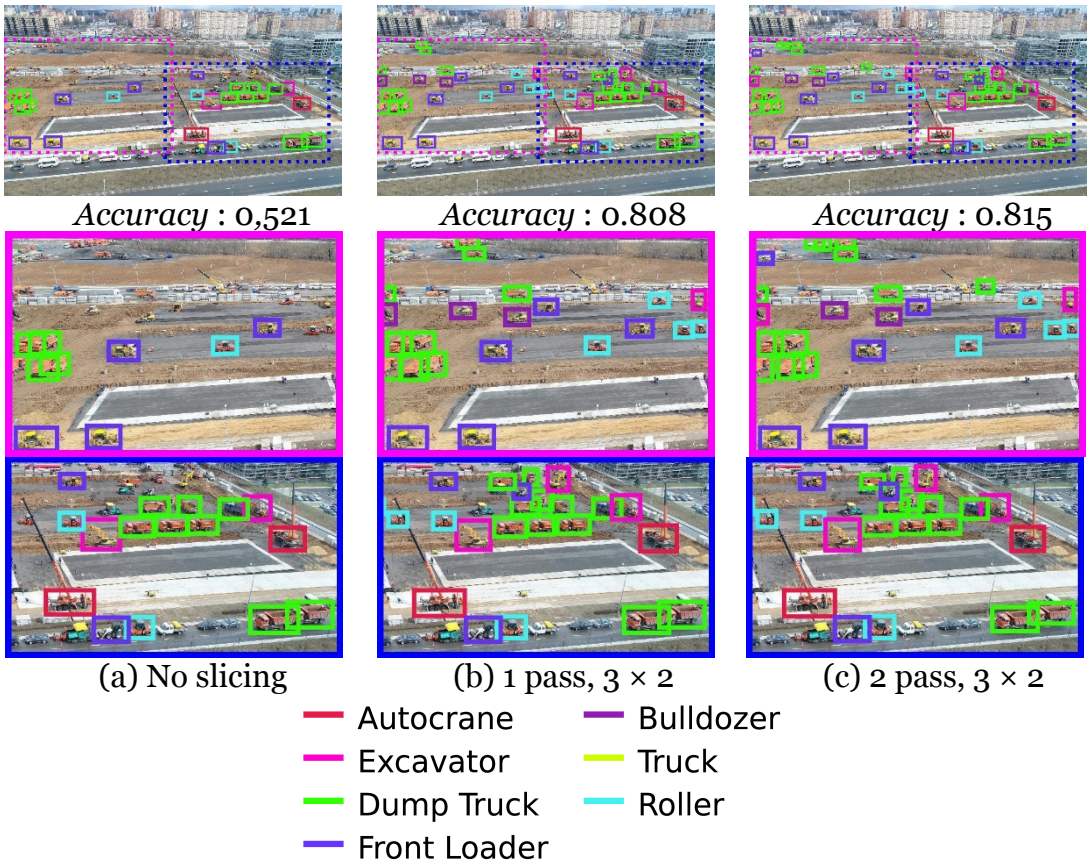


Figure 13. Detection results for (a) algorithm without slicing, (b) with 3×2 slicing, and (c) two-pass algorithm with initial slicing 3×2. Vertically - the main image and some enlarged parts of it (dotted on the main image).

Ground truth

Predicted	Autocrane	Excavator	Dump truck	Front loader	Bulldozer	Truck	Roller	Background
Autocrane	45 (68.18%)	0	0	1 (1.25%)	0	0	0	4
Excavator	0	88 (57.14%)	0	0	0	0	0	10
Dump truck	0	0	222 (52.98%)	0	0	0	0	10
Front loader	0	0	0	37 (46.25%)	1 (1.64%)	0	0	4
Bulldozer	0	0	0	1 (1.25%)	33 (54.10%)	0	0	1
Truck	0	0	0	0	0	4 (80.00%)	0	1
Roller	0	0	2 (0.48%)	0	0	0	42 (50.60%)	1
Background	21(31.82%)	66(42.86%)	195(46.54%)	41(51.25%)	27(44.26%)	1(20.00%)	41(49.40%)	0

(a) No slicing

Ground truth

Predicted	Autocrane	Excavator	Dump truck	Front loader	Bulldozer	Truck	Roller	Background
Autocrane	52 (78.79%)	2 (1.30%)	0	1 (1.25%)	0	0	0	10
Excavator	1 (1.52%)	130 (84.42%)	1 (0.24%)	0	0	0	0	42
Dump truck	0	0	344 (82.10%)	1 (1.25%)	0	0	0	35
Front loader	0	0	0	57 (71.25%)	1 (1.64%)	0	1 (1.20%)	12
Bulldozer	0	0	1 (0.24%)	2 (2.50%)	53 (86.89%)	0	1 (1.20%)	7
Truck	0	0	0	0	0	5 (100.00%)	0	3
Roller	0	0	0	1 (1.25%)	1 (1.64%)	0	67 (80.72%)	6
Background	13 (19.70%)	22 (14.29%)	73 (17.42%)	18 (22.50%)	6 (9.84%)	0	14 (16.87%)	0

(b) 1 pass, 3 × 2 slicing

Ground truth

Predicted	Autocrane	Excavator	Dump truck	Front loader	Bulldozer	Truck	Roller	Background
Autocrane	56 (84.85%)	0	0	0	0	0	0	6
Excavator	2 (3.03%)	130 (84.42%)	0	0	0	0	0	18
Dump truck	0	0	360 (85.92%)	0	0	0	0	24
Front loader	0	0	0	65 (81.25%)	2 (3.28%)	0	0	9
Bulldozer	0	0	0	2 (2.50%)	52 (85.25%)	0	0	3
Truck	0	0	0	0	0	5 (100.00%)	0	2
Roller	0	0	1 (0.24%)	1 (1.25%)	0	0	80 (96.39%)	4
Background	8 (12.12%)	24 (15.58%)	58 (13.84%)	12 (15.00%)	7 (11.48%)	0	3 (3.61%)	0

(c) 2 pass, 3 × 2

Figure 14. Confusion matrices for (a) algorithm without slicing, (b) with 3×2 slicing, and (c) two-pass algorithm with initial slicing 3 × 2.

Table 1. Statistics on the accuracy of prediction by the mAP metric on the entire dataset and its individual parts. Each part is characterized by the size of the objects presented in it - large, small and diverse.

Method	Object size			
	Large	Small	Diverse	All
1 pass, no slicing	0.905	0.271	0.712	0.55
1 pass, 3 × 2 slicing	0.84	0.696	0.801	0.739
2 pass, initial slicing 3× 2	0.928	0.783	0.823	0.835

5. Conclusion

The paper presents the idea of building a model to improve the quality of detection and bounding box visualization of multi-scale vehicles at a construction site. The FSSD512 architecture was used as the basis for the model architecture. On the first pass, the presented model used a truncated version of the SSD detector to determine the characteristic sizes of objects and set the optimal image slicing parameters for better detection on the second pass. As examples of the application of the model, cases of detection of large objects, small objects and objects presented on different plans were considered. In all the examples presented, the model achieved better accuracy than the base model and the model using slicing.

References

1. Fang, W., Ding, L., Zhong, B., Love, P. E., & Luo, H. (2018). Automated detection of workers and heavy equipment on construction sites: A convolutional neural network approach. *Advanced Engineering Informatics*, 37, 139-149.
2. Arabi, S., Haghighat, A., & Sharma, A. (2020). A deep-learning-based computer vision solution for construction vehicle detection. Computer-Aided Civil and Infrastructure Engineering, 35(7), 753-767.
3. Kim, H., Kim, H., Hong, Y. W., & Byun, H. (2018). Detecting construction equipment using a region-based fully convolutional network and transfer learning. Journal of computing in Civil Engineering, 32(2), 04017082.

4. Fang, W., Ding, L., Zhong, B., Love, P. E., & Luo, H. (2018). Automated detection of workers and heavy equipment on construction sites: A convolutional neural network approach. Advanced Engineering Informatics, 37, 139-149.
5. Zhao, Z. Q., Zheng, P., Xu, S. T., & Wu, X. (2019). Object detection with deep learning: A review. IEEE transactions on neural networks and learning systems, 30(11), 3212-3232.
6. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016). Ssd: Single shot multibox detector. In European conference on computer vision (pp. 21-37). Springer, Cham.
7. Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 779-788).
8. Deng, L., Yang, M., Li, T., He, Y., & Wang, C. (2019). RFBNet: deep multimodal networks with residual fusion blocks for RGB-D semantic segmentation. arXiv preprint arXiv:1907.00135.
9. Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. Advances in neural information processing systems, 28.
10. He, K., Gkioxari, G., Doll´ar, P., & Girshick, R. (2017). Mask r-cnn. In Proceedings of the IEEE international conference on computer vision (pp. 29612969).
11. Xu, H., Jiang, C., Liang, X., Lin, L., & Li, Z. (2019). Reasoning-rcnn: Unifying adaptive global reasoning into large-scale object detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 6419-6428).
12. Li, Z., & Zhou, F. (2017). FSSD: feature fusion single shot multibox detector.
13. arXiv preprint arXiv:1712.00960.
14. Ozge Unel, F., Ozkalayci, B. O., & Cigla, C. (2019). The power of tiling for small object detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (pp. 0-0).
15. Everingham, M., Van Gool, L., Williams, C. K., Winn, J., & Zisserman, A. (2010). The pascal visual object classes (voc) challenge. International journal of computer vision, 88(2) (pp. 303–338).