Machine Vision for Construction Equipment by Transfer Learning with Scale Models

Carl Borngrund, Ulf Bodin, Fredrik Sandin
Embedded Intelligent Systems Lab (EISLAB)
Luleå University of Technology, 971 87 Luleå, Sweden
carl.borngrund@ltu.se

***Abstract*—Machine vision is required by autonomous heavy construction equipment to navigate and interact with the environment. Wheel loaders need the ability to identify different objects and other equipment to perform the task of automatically loading and dumping material on dump trucks, which can be achieved using deep neural networks. Training such networks from scratch requires the iterative collection of potentially large amounts of video data, which is challenging at construction sites because of the complexity of safely operating heavy equipment in realistic environments. Transfer learning, for which pretrained neural networks can be retrained for use at construction sites, is thus attractive, especially if data can be acquired without full-scale experiments. We investigate the possibility of using scale-model data for training and validating two different pretrained networks and use real-world test data to examine their generalization capability. A dataset containing 268 images of a 1:16 scale model of a Volvo A60H dump truck is provided, as well as 64 test images of a full-size Volvo A25G dump truck. The code and dataset are publicly available[1]. The networks, both pretrained on the MS-COCO dataset, were fine-tuned to the created dataset, and the results indicate that both networks can learn the features of the scale-model dump truck (validation mAP of 0.82 for YOLOv3 and 0.95 for RetinaNet). Both networks can transfer these learned features to detect objects on a full-size dump truck with no additional training (test mAP of 0.70 for YOLOv3 and 0.79 for RetinaNet).**

***Index Terms*—construction equipment, automation, computer vision, deep learning, machine learning**

I. Introduction

Wheel loaders are multi-purpose vehicles often used to move materials from the ground into the tipping body of a dump truck. Dump trucks are used to move large volumes of materials from one place to another. For autonomous construction equipment to be able to function in environments where autonomous vehicles are interacting with non-autonomous vehicles and people, it is crucial that the autonomous vehicle is equipped with a reliable visual system [2].

Systems for machine vision have been developed for autonomous driving, where vehicles need to gain information from road signs, avoid obstacles and ensure that they stay on the road. Compared to autonomous construction equipment, the task of autonomous vehicles on, for example, a highway is more passive, as the vehicle is not directly interacting with the environment but rather navigating through it. Autonomous construction equipment such as wheel loaders not only navigates through an environment but also interacts with it by, for example, moving dirt around. This type of interaction makes the automation of construction equipment challenging [3].

To perform tasks that require interaction, a visual system that relies on cameras as the main sensor is needed [3]. One task in which operators rely on the visual system is, for example, when determining the point of attack on a pile when filling the bucket with material. The shape of the pile changes over time as material is removed, which typically causes the best point of attack to be different even between two consecutive scoops. Another example is when offloading material from the bucket to the tipping body of the dump truck. A visual system is needed to detect how the material inside the tipping body is positioned to decide where to offload the next bucket.

Compared to other types of visual sensors, such as radio detection and ranging (RADAR) and light detection and ranging (LIDAR), cameras have the added benefit of being relatively cheap sensors, and each frame contains a large amount of information. Although cameras are less efficient than RADAR and LIDAR in detecting distances, they are an attractive option for the automation of construction equipment due to their lower cost, high resolution and ability to distinguish objects with different surface properties, e.g., those made possible with deep neural networks.

The aim of this work is to investigate whether it is possible to use scale models in the training of a deep neural network and to use the features learned during training to detect the wheels, cab and tipping body of a full-size dump truck. When training such networks using scale models, motivated by the simplicity of the experiments compared to a full-scale approach, a key issue is how well the resulting network generalizes to the real world. In this paper, we examine the generalization properties of two different deep learning models by comparing the mean average precision (mAP) of YOLOv3 and RetinaNet in scale-model and full-scale experiments.

When off-loading materials from the bucket of a wheel loader on to the tipping body of a dump truck, it is vital that the network understands more about how the dump truck is composed of its parts rather than seeing it as one entity. It is, for example, important to understand where the tipping body

This research was conducted with support from Sweden's Innovation Agency and the ALDEE project under grant agreement no. 2019-03073 together with the Arrowhead Tools under grant agreement no. 826452 (ECSEL Joint Undertaking).

[1]The dataset can be found at [1]. The code can be found at https://github.com/phnk/yolov3 and https://github.com/phnk/keras-retinanet

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

and cab are located to make it possible for the machine to dump materials into only the tipping body and avoid dumping material over the cab. This property is especially important if autonomous vehicles are to interact with human-driven load carriers.

Our results indicate that it is possible to use the features learned by a network pretrained on images containing scale-model construction equipment to detect the corresponding objects on full-size construction equipment. However, we obtain different average precisions for the wheels, tipping body and cab and note that transfer learning from the scale-model environment is more successful for scaled-down objects that more closely resemble full-size objects.

II. Comparison of experimental environments

The process to automate earth-moving equipment can be divided in steps [3], including assisted tele-operation and fully autonomous operation. In this process, many experiments with heavy equipment partially and subsequently fully controlled by machine learning models such as neural networks need to be carried out to develop potential solutions and compare them in different environments. Thus, aspects such as the ability to accurately repeated experiments, to what extent the results generalize to realistic operational environments, the cost of the experiments, the safety and the execution time are highly relevant metrics when performing the experiments. In Table I, we qualitatively compare four different kinds of experimental environments from the perspectives of advanced simulations, basic simulations, reality and scale models when used to develop object detection features for the automation of construction equipment. Repeatability determines to what extent an experiment can be repeated with the same result. Generalization refers to how well results generalize to realistic operational conditions. Cost includes the amount of labour required by the experiment, as well as the cost of the equipment and materials. Safety concerns the risks involved when performing experiments, and the execution time deals with the experimental runtime. It should be noted that these metrics are not independent of each other. For example, if the experiment is dangerous, then the cost of maintaining constant oversight of the experiment will rise.

Real-world experiments offer the highest chance for good generalization, as these experiments are conducted in the target environment. However, the environment has properties that change over time, such as lightning and material properties, which affect the visual appearance and dynamic properties of objects. Real-world experiments also require constant supervision in case intervention is needed. To address this problem, different *simulated environments* are used to learn and generalize to the real world. Recently, simulations have shown great progress in tasks, such as computer vision and robotics tasks, where the aim was to perform simulation-to-real transfer [4]–[7] using high-end simulators [8]–[10]. It is easier to run experiments within a simulator than in the real world. If a simulator exists for the use case, as it does for autonomous driving [8], it is quite inexpensive to run experiments; however, if no such simulator exists, it is quite a costly process to create one. To our knowledge, no simulator to test autonomous construction equipment in the correct environment exists, which would mean that using simulations to train a network in the described use case would require thes implementation of such a simulator or try to use an autonomous vehicle simulator, both of which are very time consuming. Different simulators have also managed, with varying success, to simulate real-world properties such as sensor and kinetic properties [11]. Simulators are also different in how advanced they are in how they deal with realistic graphics, physics and decision making of agents [11]. A simulator that closely models real-world properties usually has an increased execution time. Examples of this phenomenon are the difficulties that exist when modelling bucket and pile interactions [12], [13].

As both simulation and real-world experiments have different strengths and weaknesses, we wanted to see if it was possible to use *scale models* to perform experiments and then generalize what the network learned to the real world. The task of object detection was used to test the viability of such a method, as object detection is important in the given use case. Scale models in this case are scaled-down versions of different construction equipment. When using scale models, the safety is high, but the execution time is still long. How well the network generalizes to a full-size environment is what this paper investigates.

III. Dataset

To investigate the possibility of using scale models of construction equipment for the (initial) training of a machine vision system, data had to be collected and labelled.

A. Data collection

The training data were collected by using a remote-controlled (RC) wheel loader scale model of comparable scale equipped with a Foxeer box camera [14] mounted on the cab. The camera recorded in 1080 p/30 fps with all the other factory settings. The same camera was used to collect the test data with full-size dump trucks.

A 1:16 scale model of a Volvo A60H dump truck was positioned in different environments, the RC wheel loader was driven towards the side of the dump truck at different angles, and the approach was recorded. To add some variance to the training and validation sets, the data were collected in different environments and under different lightning conditions. A few of these conditions can be seen in Fig. 1.

The dataset was created to encapsulate both a few different light conditions and a few different ground materials. The lightning conditions were direct sunlight, indirect sunlight and artificial light. The different ground conditions were snow, grass, gravel and pavement. The tipping body of the dump truck and the bucket of the wheel loader were also placed in both the down position and up position, as seen in Fig. 1.

The test videos, containing a full-size Volvo A25G dump truck, were collected in a similar way, where a wheel loader was allowed to approach the dump truck from the side.

TABLE I: Qualitative comparison of environments for development and validation of artificial intelligence-based automation solutions for construction equipment. Here, we investigate the generalization properties of deep neural networks for machine vision trained on a scale model.

Method	Repeatability	Generalization	Cost	Safety	Execution time
Advanced simulations	HIGH	MEDIUM	HIGH	HIGH	MEDIUM
Basic simulations	HIGH	LOW	LOW	HIGH	SHORT
Scale models	MEDIUM	**?**	MEDIUM	HIGH	LONG
Reality	LOW	HIGH	HIGH	LOW	LONG

However, these data did not contain as much variety in the environments as the videos were collected at the same place and at similar times of day.

The full-scale dump truck used is model A25G, whereas the scale-model dump truck is of an older model: A60H. Using different models allows the object we are trying to classify to vary by different amounts. In this case, the wheels are similar, the tipping body has small detail differences but still similar, and the cab has large differences, as shown in Figs. 1 and 2.

The videos used to create the training and validation set were collected in [15]; however, because of inconsistencies in which frames were extracted from the videos and how they were split into training and validation images, the dataset was recreated for this paper. A few examples of images in the dataset can be seen in Fig. 1.

B. Dataset

The collected videos were split in such a way that no frames from one video could be used for both training and validation. Each video frame was extracted at a rate of one frame every two seconds to decrease the number of similar-looking frames in the dataset. After the frames were extracted into training and validation images, the images were labelled using the YOLOv3 label structure [16]. The test data were extracted from the test videos and labelled in the same way. The dataset contained 268 training images, 90 validation images and 64 test images. A few examples of the training images can be seen in 1. Only the central dump truck was labelled, and we assume that when we are looking for wheels, cabs and tipping bodies, we are only looking for dump truck wheels, cabs and tipping bodies. As neither vehicle in the background is a dump truck, all detections on any of these vehicles will be treated as false positives.

From Fig. 1c and 1d, it is important to realize that the bucket of the wheel loader can block the entire view of, for example, the back wheels. This realization is important, as the bucket can block the direct view of the tipping body while offloading materials from the bucket, which is a concern that needs to be addressed.

IV. Experimental set-up

A. Networks

The networks used in these experiments were YOLOv3 [17] and RetinaNet [18]. Two networks were chosen, not to compare them against each other but rather to make sure that different networks can learn similar features. We felt that it was important to see whether or not two completely different networks could learn and perform well to make sure that the results are not dependent on a single network.

1) YOLOv3: YOLOv3 [17] is an object detector that was first introduced in 2015 [19] to show that it was possible to achieve real-time performance while still maintaining high accuracy. As the name alludes, YOLOv3 is the third version of YOLO, or You Only Look Once, which has incrementally improved over the last few years. The implementation of YOLOv3 used was that from [16], and it was implemented using PyTorch [20]. When training this implementation on a dataset other than COCO [21], it is recommended by the author to change the internal filters in YOLOv3 to match the number of classes in the dataset. YOLOv3 uses Darknet-53 as its backbone network [22].

2) RetinaNet: The second network used was RetinaNet [18]. The aim of RetinaNet was to present a one-stage detector that matches the current state-of-the-art performance of two-stage detectors on the COCO dataset. This network was introduced after YOLO9000 (v2) [22] in early 2018. The RetinaNet implementation used can be found at [23] and was implemented using Keras [24]. No implementation changes were made to this network, nor any configuration. RetinaNet uses ResNet-50 [25] as its backbone network.

B. Training set-up

When training both networks, the same hyperparameters were used where possible; however, no large changes were made to the implementation of either training schema, which includes no changes to the optimizer, learning rate scheduler, loss function or the size of the input images. The hyperparameters used during training for both networks were 500 epochs, a batch size of 1, 268 steps and a learning rate of 0.001. As there were 268 training images in the dataset, using a batch size of 1 and 268 steps ensured that 1 epoch means that we looked through the entire dataset only once. These are hyperparameters that can be matched easily between the two networks, but as mentioned, hyperparameters that would have required a lot of work were not changed. These differences are described below. Both networks were pretrained on the COCO dataset and then fine-tuned on the given dataset.

In this context, fine-tuning means that we let all weights be updated during the training phase of both networks.

1) YOLOv3: YOLOv3 was trained on images with a width of 416 pixels and a height of 416 pixels (the default size). We used the loss function defined in [19] and SGD as an optimizer with a short burn-in phase at the start of training, which ramps up the learning rate over the first 1000 iterations. This

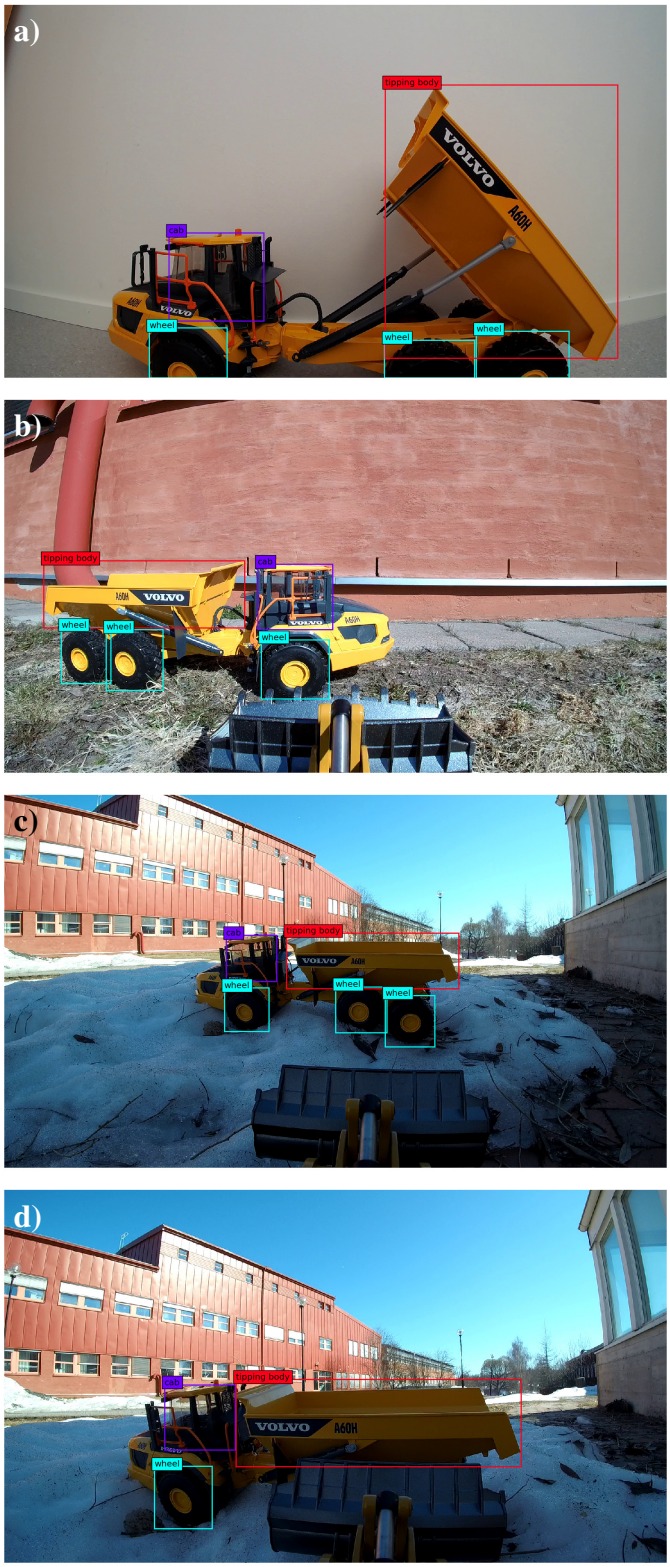


Fig. 1: A few example images from the training set on which the targets have been drawn. a) The dump truck positioned indoors with an artificial light source. b) The dump truck positioned on grass with direct sunlight. c) The dump truck positioned on snow in the shadow. d) The bucket of the wheel loader blocks the view of two wheels.

Fig. 2: Example of an image from the full-scale test data with a wheel loader that approaches the dump truck from the side.

procedure was done to help keep the network from diverging. To help the network generalize, a few augmentation techniques were utilized: translation, rotation, shear, scaling, reflection, HSV saturation, HSV intensity and quality degradation. The images were augmented at a rate of 50%, or in other words, every other image was augmented in some way.

2) RetinaNet: RetinaNet was trained on images with a width of 1333 pixels and a height of 800 pixels (the default size). The implemented RetinaNet [23] used the focal loss for the classification loss and smooth_l1 for the bounding box regression loss. RetinaNet used the Adam optimizer and reduced the learning rate whenever the validation loss plateaued. The augmentation techniques utilized in the implementation of RetinaNet were rotation, translation, shearing, scaling, horizontal flipping and vertical flipping.

Another RetinaNet network was also trained on images of size 416x416 to see whether it could match YOLOv3's performance.

C. Testing set-up

The metrics used to measure how well the object detector was performing are the average precision (AP), mean average precision (mAP) and inference time. AP and mAP are used to measure how well the network performed, and the inference time is used to determine whether the network can meet the real-time requirement of 0.033 seconds per frame. The inference time was calculated by taking the average inference time of the different networks when processing every frame of one test video containing the full-size construction equipment. AP was calculated as described in [26], where the precision, p, and the recall, r, were calculated by $p = \frac{TP}{TP+FP}$ and $r = \frac{TP}{TP+FP}$. mAP was calculated as the average over all the classes of AP values. Both networks used an intersection-over-union (IoU) threshold of 0.50, confidence threshold of 0.55 and a non-maximum suppression (NMS) threshold of 0.50. Examples of the test images can be seen in Fig. 2.

Both networks' test image input size was decreased incrementally to see how the two networks perform in regards

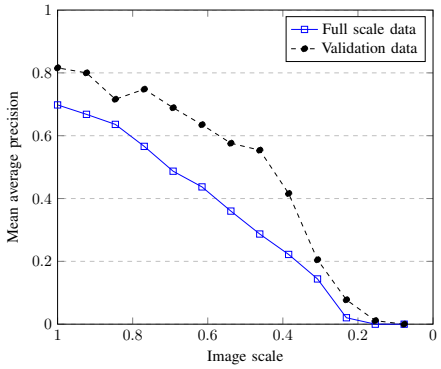


Fig. 3: YOLOv3 image scale vs the mean average precision. Each marker is a data point. Here, an image scale of 1 denotes images with the default size of 416x416, and smaller scales denote scaled-down images resulting in objects with lower resolution.

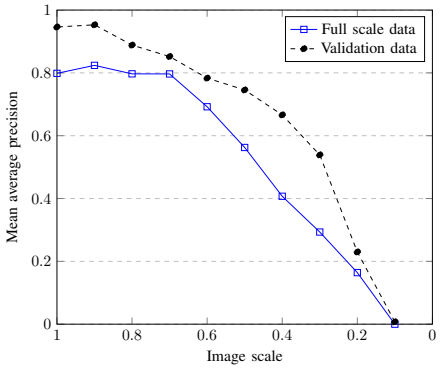


Fig. 4: RetinaNet image scale vs the mean average precision. Each marker is a data point. Here, a scale of 1 denotes images of the default size of 1333x800, and smaller scales denote scaled-down images.

to mAP. The implementation of YOLOv3 required that input images be a multiple of 32; thus, in every trial, the input image width and height were both decreased by 32. RetinaNet, however, did not have such a limitation; the width and height of the images in RetinaNet were decreased by 10% per trial, which translates to a decrease in the width of $\lfloor 0.10 * 1333 \rfloor = 133$ and a decrease in the height of 80.

Images larger than the training images were not tested. For images smaller than the input, we can simulate the fact that the objects we are trying to detect are farther away, and we can obtain an understanding of how the networks will perform on smaller objects.

V. Results

As mentioned in Section IV-C, the metrics used in this paper are the inference time, AP and mAP. Section IV-C also discussed that both networks used an IoU threshold of 0.50, a confidence threshold of 0.55 and an NMS threshold of 0.50. A few images with bounding boxes from the two networks can be seen in Fig. 7. Both the training and inference tests were conducted using an Nvidia 980 Ti [27]. The data collection and labelling procedures are described in Sections III-A and III-B, respectively. After training, the validation mAP on images with the trained input size was 0.82 for YOLOv3 and 0.95 for RetinaNet. RetinaNet, which was trained on 416x416 images, performed fine on the validation data but had a hard time transferring the knowledge to the test data. It reached an mAP of 0.72 on the validation set and 0.38 on the test set.

Table II shows the test results for both networks using the same size of input images as those used during training. As RetinaNet is trained on larger images, it will take longer to perform inference compared to YOLOv3. Both networks' performances are comparable to each other; however, RetinaNet has a larger difference between the precision and recall. Table II indicates that both YOLOv3 and RetinaNet can transfer the features of the wheels from a scale-model dump truck to a full-size dump truck. It also indicates that transferring the features for the cab is very difficult for RetinaNet, while transferring features about the tipping body is easier. YOLOv3 outperforms RetinaNet in detecting the cab, while RetinaNet outperforms YOLOv3 in detecting the tipping body. Both networks do, however, have a harder time finding the cab than any other part of the dump truck. The low score in the cab class for RetinaNet may also explain why the RetinaNet recall is so low. Figs. 5c and 6c reinforce the finding that both networks can learn the features of the cab in the training images but have a hard time transferring that knowledge to the full-size cab regardless of the input image size.

As discussed in Section IV-C, the width and height of the input images were decreased incrementally for each data point. The values of the width and height were then divided by the original size of the training images to be presented in Figs. 3, 4, 5 and 6, which means that the image scale cannot be compared between the two networks, as they do not mean that the networks have the same input size but rather that the input has been decreased by the same percentage.

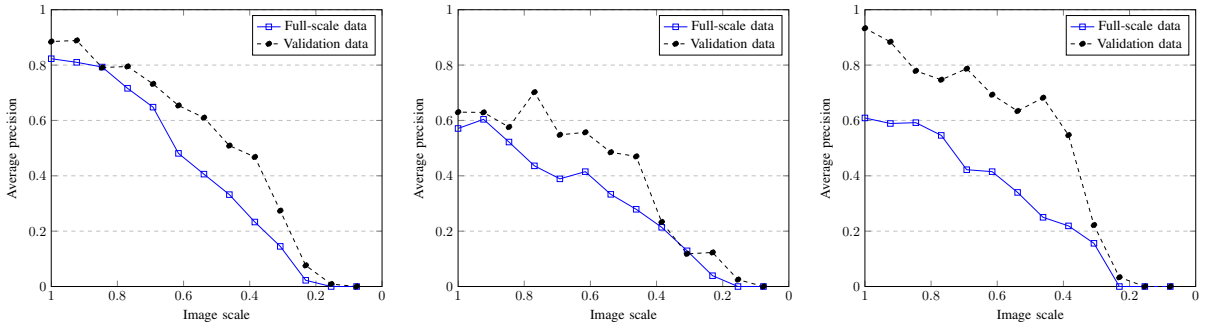
Both Figs. 4 and 3 show that both networks can learn the features in the training set and use them to understand the test set, and the difference between the validation mAP and test mAP is similar for both networks.

From Fig. 3, we can see that YOLOv3 always has decreased performance when the input is down-scaled. However, from Fig. 4, the RetinaNet's test mAP seems to start on a plateau of an mAP of approximately 0.8 but drops off at a scale of 0.6 of the training size. The RetinaNet validation mAP is always decreasing, however. The fact that the test mAP for RetinaNet starts off plateauing indicates that an image decreased to 70% of the size of the original input image will have minimal impact on RetinaNet's performance.

From Figs. 5a, 5b, 6a and 6b, we can see that both networks have similar performance on the test data compared to the validation data for the wheel and tipping body classes. From Figs. 5c and 6c, we can see that both networks have a large discrepancy between the validation AP and test AP for the cab class. This discrepancy seems to be size independent except

TABLE II: Test results of the two networks for the default image sizes.

Network	Size	Precision	Recall	Wheel AP	Cab AP	Tipping body AP	mAP	Inference time
YOLOv3	416x416	0.74	0.77	0.83	0.59	0.67	0.70	0.029
RetinaNet	1333x800	0.80	0.49	0.89	0.29	0.93	0.79	0.098

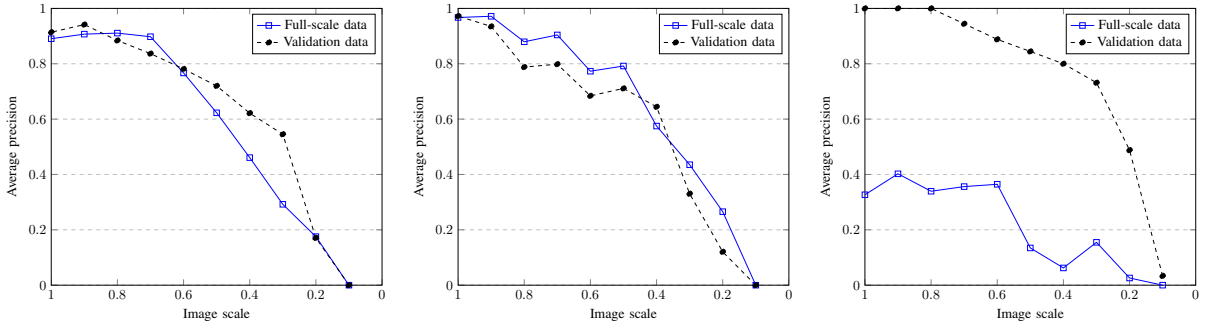


(a) YOLOv3 wheel AP validation vs testing.

(b) YOLOv3 tipping body AP validation vs testing.

(c) YOLOv3 cab AP validation vs testing.

Fig. 5: Individual class AP for each class using YOLOv3. The AP is used as we are looking at the performance per class.



(a) RetinaNet wheel AP validation vs testing.

(b) RetinaNet tipping body AP validation vs testing.

(c) RetinaNet cab AP validation vs testing.

Fig. 6: Individual class AP for each class using RetinaNet. The validation vs the test AP.

for very small images where the performance is very low. YOLOv3 does, however, perform better on the cab class for the test images and decreases the performance on the cab class similarly to the other classes in the dataset. RetinaNet does not seem to be able to understand the features of the full-size cab regardless of the input size.

As discussed in Section IV-C, if we consider reduced images as simulating smaller objects in the images, we can see that the networks will perform much worse on smaller images, as shown in Figs. 3 and 4. For example, if all the objects in the image are 40% smaller, then RetinaNet will lose approximately 50% accuracy.

VI. Discussion

Overall, both networks learned the features in the training set, performed well on the validation set and could generalize the features learned during training to the test set, which can be seen in Figs. 3 and 4. There was no class that was impossible for the networks to learn on the scale-model dump truck during training. From Fig. 5, we can see that YOLOv3 has a similar mAP for the wheel and tipping body class during both validation and testing, while it performs much worse on detecting the cab class. Figs. 4 and 6 show that RetinaNet produces similar results, as it performs quite well on the wheel and tipping body classes but has a much lower performance on the cab class.

From the inference examples in Fig. 7, we can see that neither network is perfect. All 4 sub-figures contain false positives of some sort, while Figs. 7a and 1d both contain false negatives. When the wheel loader is far away from the dump truck, all the detected objects are very small. As seen from Figs. 3 and 3, both networks have a lower performance when trying to find small objects.

Moreover, to be able to reach level (5) of automating earth-moving equipment as described in Section I, it is very important to understand what type of representation a planning system needs from the perception system. Many papers use world models [28], [29], [30] to describe the internal representation of all systems, which can help the planning system perform. Systems other than the perception system on a wheel

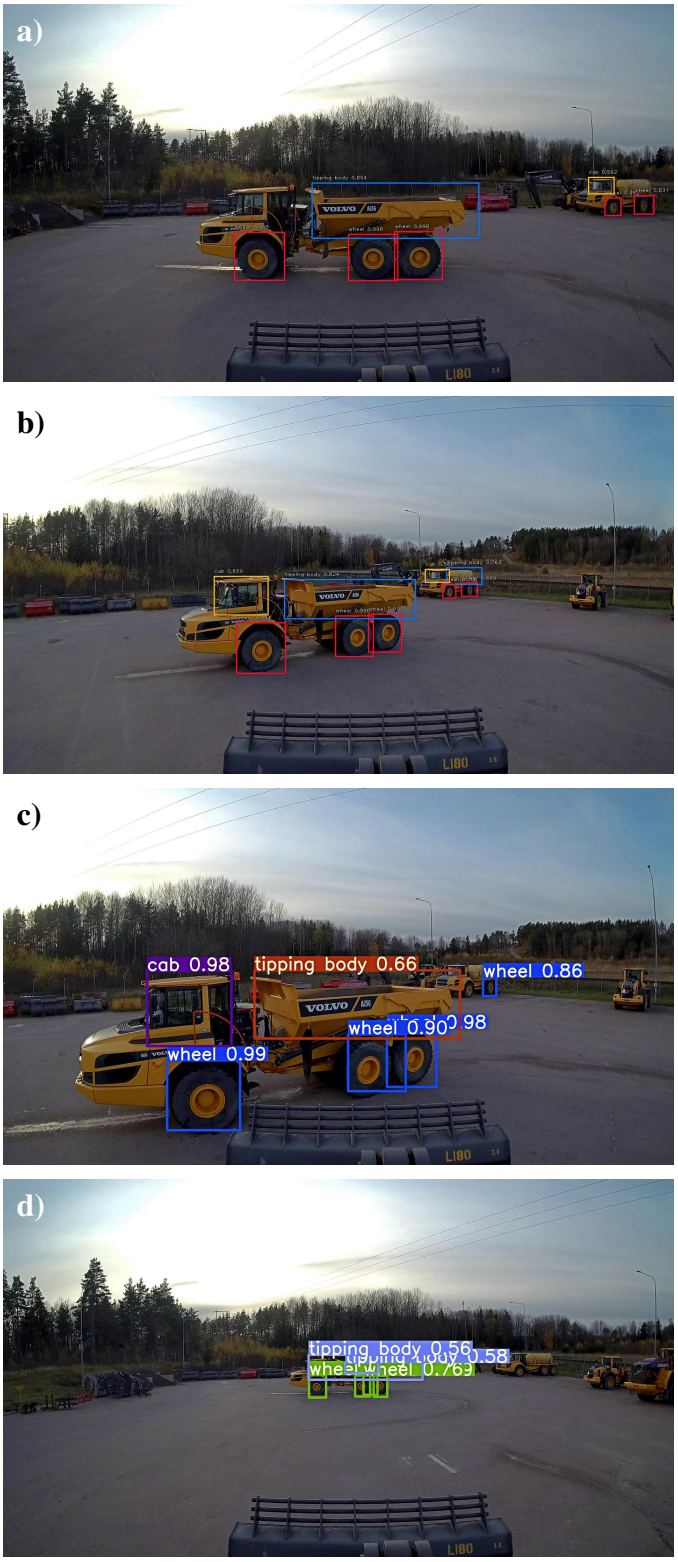


Fig. 7: A few inference examples from both YOLOv3 and RetinaNet. a) Inference example of RetinaNet where the network has difficulties detecting the cab class. b) Inference example of RetinaNet where all the objects were correctly detected and classified. c) Inference example of YOLOv3 where all dump truck objects were correctly detected and classified. One wheel of a distant vehicle in the background is also detected. d) Inference example of YOLOv3 from very far away. YOLOv3 has a hard time with smaller objects that are far away. A false negative is in the cab class. There are two false positives: one in the tipping body class and one in the wheel class.

loader could be all other perception systems on all the other vehicles, a mapping system or other sensor information. These systems can all be fused to give the wheel loader a better understanding of its surroundings and help the planning system perform well.

For example, when the wheel loader is far away but has the bucket filled with material, it is still critical that the networks can identify the dump truck that is ready for the material and ignore other vehicles that exist in the background. To address this problem, we could take inspiration from the human visual system where humans have good resolution on the focal point, which allows for more resources to be used to process what is near the focal point.

Another example might be that once the wheel loader is close to the correct dump truck, it is vital that the networks can correctly identify both the cab and the tipping body, as dumping materials over the cab can result in serious bodily harm or death to the driver of the dump truck. To combat this problem, the system could use other high-level information about dump trucks, such as the two back wheels being always above the tipping body to help the system reassure itself that it is in fact performing a safe unloading of materials into the tipping body.

As both networks perform well on the cab during validation but poorly during testing, as seen in Figs. 5c and 6c, it leads us to believe that we do not have enough training and validation data, the training and validation data do not contain enough variance or that the differences in the cab between the scale-model as seen in Figs. 1 and 2 are too large. We assume that these hypotheses are true, but we cannot confirm them as there exists no exact scale-model copy of the A25G dump truck. Another reason why the wheels are easy to detect while the cab is difficult to detect might be because the wheels are very distinct and look similar regardless of the lighting conditions or approach angle, while the look of the cab changes considerably when the lighting conditions and approach angle change. The cab is also quite different between the scale-model and the full-size dump truck, where the plastic of the scale-model dump truck does not reflect light in the same way as the windows of the full-size dump truck.

From Figs. 6a and 6b, we can see that RetinaNet performs better on the test data than the validation data for some image scales. As the tipping body can be in an up position, as shown in Fig. 1a, the validation data add more complexity to both the tipping body and the wheels. The wheel class becomes harder to recognize because it is now possible to see the wheels behind the tipping body.

VII. Conclusion and Future Work

As mentioned in Sections I and II, working with large construction equipment is an expensive and time-consuming process. Simulations have their own drawbacks, and it can be difficult to properly model camera or kinetic processes. In this paper, we tried to determine whether using scale models for training and testing on real construction equipment could fill this gap. Two different pretrained object detection networks,

YOLOv3 [17] and RetinaNet [18], were trained and evaluated on a labelled dataset including images of both scale-model and full-size construction equipment. The training and validation sets contained images of scale-model construction equipment, and the test set contained images of full-size construction equipment. The results indicate that it is possible to use the features learned by the networks during training and transfer them to test data with no additional training. The results also indicate that objects that are more similar between the scale-model and full-size dump truck are easier than objects that are very different. The generalization score for the scale models as described in Table I would thus be medium. YOLOv3 reached a validation mAP of 0.82 and a test mAP of 0.70, while RetinaNet reached a validation mAP of 0.95 and a test mAP of 0.79.

The results obtained in this work pave the way for continued work on studying the use of scale models for the development of AI solutions for the autonomous operation of construction equipment. Our goal is to investigate whether and how scale models can be used for the development of a navigation system that uses camera-based machine vision to autonomously navigate the wheel loader towards the dump truck tipping body and correctly position it for offloading materials in construction environments.

Future work should investigate kinetic property transfer, as the scale models weigh much less than the real construction equipment, which will help contribute to the larger goal of this paper.

Acknowledgements

We thank Torbjörn Martinsson at Volvo CE for his contribution to capturing the videos of the full-size video data.

References

[1] C. Borngrund, "Dump truck object detection dataset including scale-models," diva2:1384401, 2020.

[2] J. V. Brummelen, M. O'Brien, D. Gruyer, and H. Najjaran, "Autonomous vehicle perception: The technology of today and tomorrow," *Transportation Research Part C: Emerging Technologies*, vol. 89, pp. 384 – 406, 2018.

[3] S. Dadhich, "Automation of wheel-loaders," Ph.D. dissertation, Luleå University of Technology, EISLAB, 2018.

[4] X. Yue, Y. Zhang, S. Zhao, A. Sangiovanni-Vincentelli, K. Keutzer, and B. Gong, "Domain randomization and pyramid consistency: Simulation-to-real generalization without accessing target domain data," 2019.

[5] OpenAI, M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng, and W. Zaremba, "Learning dexterous in-hand manipulation," 2018.

[6] M. Johnson-Roberson, C. Barto, R. Mehta, S. N. Sridhar, K. Rosaen, and R. Vasudevan, "Driving in the matrix: Can virtual worlds replace human-generated annotations for real world tasks?" in *IEEE International Conference on Robotics and Automation*, 2017, pp. 1–8.

[7] S. Hinterstoisser, V. Lepetit, P. Wohlhart, and K. Konolige, "On pre-trained image features and synthetic images for deep learning," in *Computer Vision – ECCV 2018 Workshops*, L. Leal-Taixé and S. Roth, Eds. Cham: Springer International Publishing, 2019, pp. 682–697.

[8] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, pp. 1–16.

[9] S. Shital, D. Debadeepta, L. Chris, and K. Ashish, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," in *Field and Service Robotics*, H. Marco and S. Roland, Eds. Springer International Publishing, 2018, pp. 621–635.

[10] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," in *Field and Service Robotics*, 2017.

[11] F. Rosique, P. Navarro Lorente, C. Fernandez, and A. Padilla, "A systematic review of perception system and simulators for autonomous vehicles research," *Sensors*, vol. 19, p. 648, 02 2019.

[12] X. Shi, P. J. A. Lever, and F. . Wang, "Fuzzy behavior integration and action fusion for robotic excavation," *IEEE Transactions on Industrial Electronics*, vol. 43, no. 3, pp. 395–402, June 1996.

[13] G. J. Maeda, "Learning and reacting with inaccurate prediction: Applications to autonomous excavation," Ph.D. dissertation, University of Sydney, 2013.

[14] "Foxeer 4k box action camera supervision," http://foxeer.com/Foxeer-4K-Box-Action-Camera-SuperVision-g-22, accessed: 2019-12-16.

[15] C. Borngrund, "Machine vision for automation of earth-moving machines : Transfer learning experiments with yolov3," Master's thesis, Luleå University of Technology, Department of Computer Science, Electrical and Space Engineering, 2019.

[16] G. Jocher, guigarfr, perry0418, Ttayu, J. Veitch-Michaelis, G. Bianconi, F. Baltacı, D. Suess, WannaSeaU, and IlyaOvodov, "ultralytics/yolov3: Rectangular Inference, Conv2d + Batchnorm2d Layer Fusion," Apr. 2019. [Online]. Available: https://doi.org/10.5281/zenodo.2672652

[17] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," 2018.

[18] T. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 2, pp. 318–327, 2018.

[19] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788.

[20] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035.

[21] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham: Springer International Publishing, 2014, pp. 740–755.

[22] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 6517–6525.

[23] H. Gaiser, M. de Vries, V. Lacatusu, vcarpani, A. Williamson, E. Liscio, András, Y. Henon, jjiun, C. Gratie, M. Morariu, C. Ye, M. Zlocha, B. Weinstein, R. M. de Andrade, P. Conceição, A. Pacha, hannesedvartsen, D. Shahrokhian, W. Fang, M. Clark, meagerYak, I. Jordal, M. V. Sande, Jin, Etienne-Meunier, A. Grigorev, G. Erhard, E. Ramos, and D. Dowling, "fizyr/keras-retinanet 0.5.1," Jun. 2019. [Online]. Available: https://doi.org/10.5281/zenodo.3250670

[24] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[25] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[26] M. Zhu, "Recall, precision and average precision," p. 11, 09 2004.

[27] "Nvidia geforce gtx 980 ti," https://www.nvidia.com/en-us/geforce/900-series/, accessed: 2019-12-16.

[28] S. Balakirsky and A. Lacaze, "World modeling and behavior generation for autonomous ground vehicle," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, vol. 2, April 2000, pp. 1201–1206 vol.2.

[29] A. Kelly, A. Stentz, O. Amidi, M. Bode, D. M. Bradley, A. Diaz-Calderon, M. Happold, H. Herman, R. Mandelbaum, T. Pilarski, P. Rander, S. Thayer, N. Vallidis, and R. Warner, "Toward reliable off road autonomous vehicles operating in challenging environments," *The International Journal of Robotics Research*, vol. 25, pp. 449 – 483, 2004.

[30] T. H. Hong, M. Abrams, T. Chang, and M. Shneier, "An intelligent world model for autonomous off-road driving," *Computer Vision and Image Understanding, Hook, S. Aster Spectral Library.*, 2000.