Hierarchical 2-step neural-based LEGO bricks detection and labeling

Tomasz BOINSKI

Gdansk University of Technology, Gdansk, Poland, tomboins@pg.edu.pl
ORCID: 0000-0001-5928-5782

Abstract

LEGO bricks are extremely popular and allow the creation of almost any type of construction due to multiple shapes available. LEGO building requires however proper brick arrangement, usually done by shape. With over 3700 different LEGO parts this can be troublesome. In this paper, we propose a solution for object detection and annotation on images. The solution is designed as a part of an automated LEGO bricks arrangement. The proposed approach consists of 2 stages – object detection and labeling. The paper discusses different approaches and points out a final model. A 2-step, hierarchical model and the results are presented. An evaluation of the proposed solution is also given.

Keywords: Data annotation, Object detection, Transfer learning.

Introduction

LEGO bricks are extremely popular and allow the creation of almost any type of construction. This can be done thanks to the multiple shapes available. Utilizing the full potential of construction possibilities requires however proper brick arrangement. The usual sorting, for average collections, is done by shape, as the colors and decals can be easily distinguished even in a big pail of bricks (Aplhin 2020). This cannot be said about the shape, and the basic problem is to find a brick of the required shape and size. With over 3700 different LEGO parts (Maren 2018) the ability to find a given part is crucial in every day constructing, especially for very large LEGO collections, where shuffling through large boxes of mixed bricks is not acceptable.

Unfortunately, there is no simple solution for brick sorting. LEGO Group provides only a simple sorting mechanism based on the brick size in form of the 2011 released LEGO Sort and Store item. Other solutions usually rely on optimizing the process of sorting rather than automating it (e.g. (Adam 2019)). The heart of the construction, in any form it may be like made of bricks itself or designed using a 3D printer, requires a mechanism for bricks detection and labeling.

With the current state of neural network development and performance of modern computers the aforementioned problem can be dealt with. Independently from the way of building the sorting machine it requires a well-trained neural network able to distinguish between different, often very similar bricks, or at least divide them into smaller categories, allowing further manual selection of proper bricks. To date, few projects took this approach, with a notable example of (West 2019). The approach is based on images generated using the LDraw library (https://www.ldraw.org/), containing 3D models of every brick available.

In this paper, we propose a two-step approach to LEGO brick detection and labeling. The structure of this paper is as follows. In the next section available approaches to object detection are presented. In the following section we propose a hierarchical solution for LEGO bricks detection and annotation. The detection and labeling stage are described in the first and second subsections respectively. Finally, a summary is given.

Approaches for classification of large number of classes

Object detection on images is a complex task. Much work has been done however in this field, especially since the emergence of deep neural networks (Erhan et al. 2014; Galvez et al. 2018; Szegedy, Toshev, and Erhan 2013). Over time general-purpose pre-trained models were established for general object detection.

One of the first such models was "Inception" (Szegedy, Liu, et al. 2014), which aimed at setting the new state of the art for classification and detection in the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) (Russakovsky et al.

Cite this Article as: Tomasz BOINSKI "Hierarchical 2-step neural-based LEGO bricks detection and labeling" Proceedings of the 37th International Business Information Management Association (IBIMA), ISBN: 978-0-9998551-6-4, 1-2 April 2021, Cordoba, Spain

2015). The authors aimed at better resources utilization and thus allowing increasing the depth and width of the network. The proposed solution ranked 1st in the 2014 edition of the challenge with mAP 43.9%.

Another proposed solution, based on similar principles is a Fast Region-based Convolutional Network method (Fast R-CNN) (Girshick 2015). The authors improved existing R-CNN (Girshick et al. 2014) solutions increasing both training and testing speed while also increasing detection accuracy. Fast R-CNN trains the very deep VGG16 (Simonyan and Zisserman 2014) network 9x faster than R-CNN, is 213x faster at test-time, and achieves a higher mAP on PASCAL VOC 2012 (Everingham et al. 2012). Compared to SPPnet, Fast R-CNN trains VGG16 3x faster, tests 10x faster, and is more accurate.

The drawback of the aforementioned solutions was a mediocre performance. The Single Shot MultiBox Detector (SSD) aimed at solving this problem (Liu et al. 2016). This approach discretizes the output space of bounding boxes into a set of default boxes over different aspect ratios and scales per feature map location. During prediction, the shape of each box is adjusted based on the score for the presence of each category in the given default box. Furthermore, the predictions are combined based on the result from multiple feature maps for different resolutions. This, compared to other single stage methods, grants SSD much better accuracy, even with a smaller input image size. The model is also faster as it eliminates proposal generation and subsequent pixel or feature resampling stage. Test on PASCAL VOC and ILSVRC datasets shows that even due to the elimination of the first stage SSD has comparable accuracy to methods.

Another solution aiming at increasing the performance of the detection was YOLO (You Only Look Once) model (Redmon et al. 2016). This approach uses only one network to assign class probabilities to spatially separated bounding boxes. This unified architecture proved to be extremely fast. The base YOLO model can process Images at 45 frames per second. A smaller version of the network, called Fast YOLO, can go up to even 155 frames per second while still achieving double the mAP of other real-time detectors. The drawback is are localization errors. The authors, however, managed to limit false detections where no object is present. The solution became widely used and is constantly upgraded by different teams, the current version is v5 (Jocher et al. 2021).

The aforementioned approaches rely on single-stage object detection and labeling. Other solutions, where the detection and the labeling stages are separated and usually rely on a combination of the aforementioned approaches. One of such approaches is (Yahalomi 2020). The solution performs object detection and annotation in 2 stages oriented hierarchically. The first stage consists of a single network that detects general classes. The second stage consists of several separate networks, that aim at refining the classification of each of the previously detected objects. This allowed improvement of the classification error by 3-5 times compared to other state-of-the-art single-stage solutions (down to 2.5-4.5% from 12%).

In this paper, we tested different approaches for usage in a hierarchical solution for LEGO bricks detection and annotation.

Classification of LEGO bricks

In our approach, we decided to perform two steps. The first step of LEGO labeling is the brick detection on the background. We decided to introduce this step to separate the bricks from other objects that might be visible on the image, like debris, damages to the conveyor belt, shadows, parts of the sorting machine, etc. This will be especially useful when trying to label objects that are very close to the border of the conveyor belt or touching the edges of the machine. During the second stage, we try to annotate the detected object with a proper label. For the brick detection stage, we took images of bricks on different backgrounds. For brick labeling, we assume that the bricks move on the white conveyor belt. In all cases, the labeling was done using labelImg software (Tzutalin 2015). By separating those two steps we aim at achieving better results as the objects for labeling are already detected as LEGO bricks thus simplifying the second stage by removing the need to eliminate unknown objects (Yahalomi 2020).

Bricks Detection

During this step, we aimed at training a network to be able to detect only 1 class - LEGO bricks. For this part, we created a data set containing 5841 images, of which 2933 were real bricks photos and the remaining 2908 were LDraw based renders (Zawora et al. 2021). The renders contain 1 brick each and the real photos contain from 1 to 32 bricks. In total, the images contained 7688 bricks, which were surrounded by bounding boxes.

The images were divided into the training and validation sets randomly. Out of the real images we randomly selected 131 for the validation set and 2802 for the training set. The images in the training set were resized to 640x640 pixels keeping the aspect ratio and then augmented with the following transformations:

- horizontal and vertical mirroring,
- randomly cutting up to 5% of the brick,

- blurring,
- increasing and decreasing contrast,
- noise introduction,
- increasing and decreasing of the brightness,
- rotation by a small degree (so as not to distort bounding boxes).

The training set contained a total of 19872 real images. To that, we added 2908 renders.

The training itself was done with 2 approaches. First we used TensorFlow Object Detection API with RetinaNet-50 (Lin et al. 2018) and EfficientDet (Tan, Pang, and Le 2020) networks already trained for object detection (Tensorflow 2020). Later on we trained the YOLOv5 (Jocher et al. 2021) based network.

Training the RetinaNet-50 network using renders and real images did not work well. As we can see in Figures 1 and 2 other objects were detected as bricks and the bricks were detected multiple times. Furthermore both the precision and recall showed a big fluctuation of values during the training process.

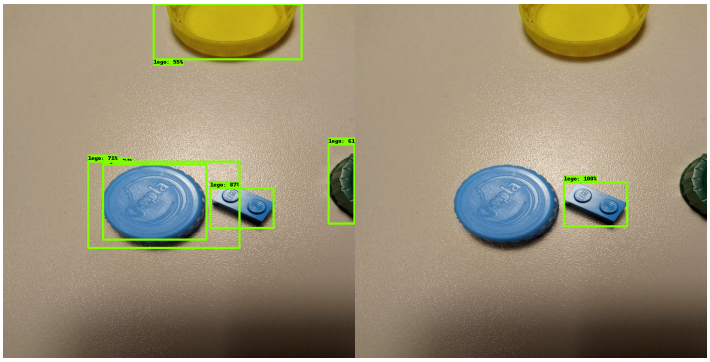

Fig. 1: Errors (wrong objects detected) in brick detection by RetinaNet-50 (left) and expected results (right).

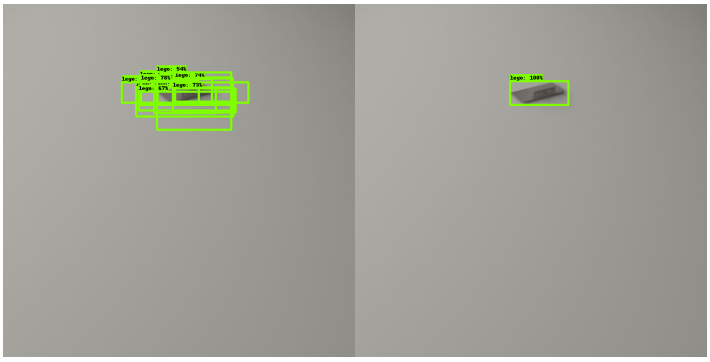

Fig. 2: Errors (multiple detections) in brick detection by RetinaNet-50 (left) and expected results (right).

We eliminated the renders from the training set as the white, constant background of the renders lead to detecting any object present on the image as a brick. This time the results were better however overall still lacked in quality and both the precision and recall did not reach values higher than 0.5.

Next, we tried EfficientDet network. This time we trained the network using only real images and discarded all renders. This time the quality was much better with higher both precision and recall. Furthermore, the values were more stable and we did not observe sudden jumps in both precision and recall values. Comparison of results between RetinaNet-50 and EfficientDet are presented in Figure 3.

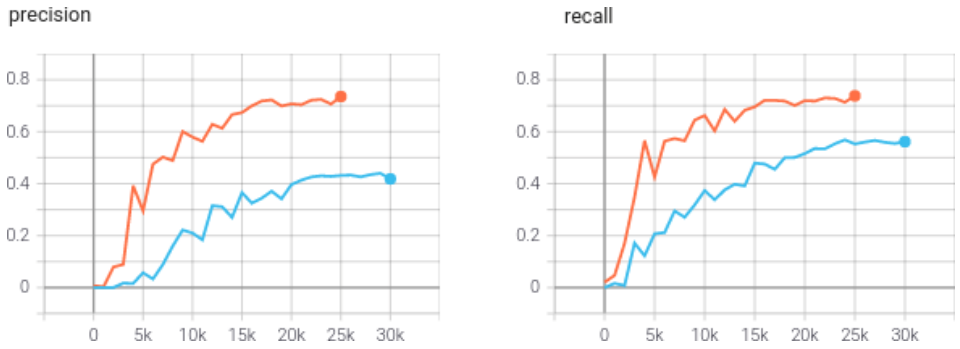

Fig. 3: Precision (left) and recall (right) comparison (RetinaNet-50 - blue, EfficientDet - red).

During the second approach, we used YOLO (You Only Look Once) version 5 network. For this network, we tested two models, small (YOLOv5s) and medium (YOLOv5m) (ultralytics 2021).

YOLOv5m proved to perform very well, with precision and recall reaching over 0.9 after the first few epochs. Further tests with YOLOv5s network showed that even though this model is much simpler and faster to train it still reaches similar levels of both precision and recall (Fig. 4).

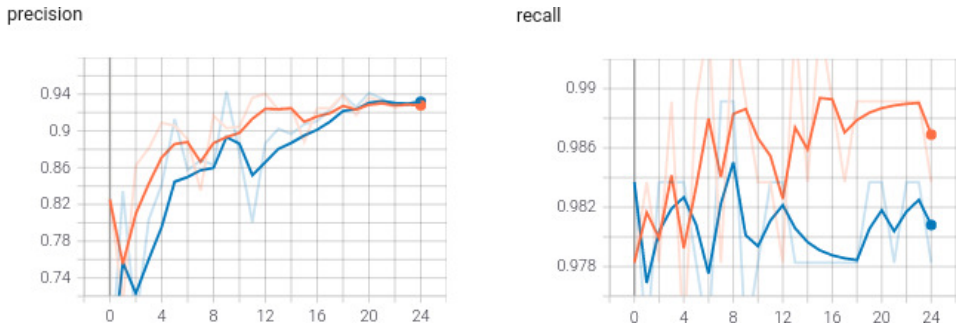

Fig. 4: Precision and recall comparison (YOLOv5s - blue, YOLOv5m - red).

As we can see YOLOv5s network proved to be superior for bricks detection and in further works, during the bricks annotation we decided to use this model. Furthermore, the training process could be shortened thanks to the usage of transfer learning (Pan and Yang 2009; Torrey and Shavlik 2010). With a more traditional approach, similar results were achieved after over a dozen of epochs. The achieved results proved to be very good. As can be seen in Fig. 5 the network was able to detect the LEGO bricks even in complicated scenarios.

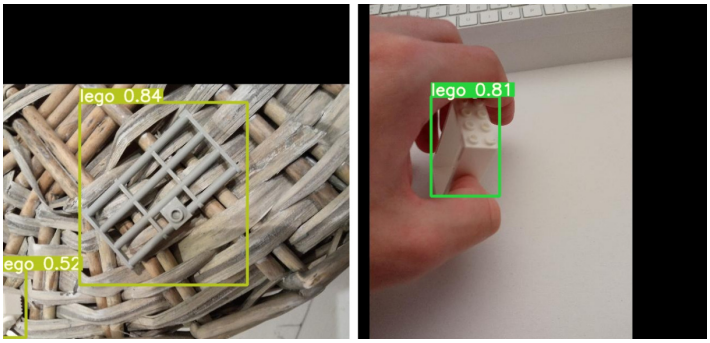

Fig. 5: Sample LEGO brick detection results.

What is worth noticing is that the model can work with multiple objects at once. This way we can quickly generate new training data by simply detecting multiple LEGO bricks at once in the camera view. Providing that all the bricks are of the same type we can thus quickly create annotated data-sets that can be further used to train the labeling network as presented in the next section.

Bricks Labeling

Having LEGO bricks separated from the background we can start working on bricks labeling. Currently, we are working with 10 classes describing different LEGO shapes (3001, 3002, 3003, 3004, 3710, 3009, 3010, 3007, 3034, and 3832 from official LEGO part numbers. Full list available at (Bricklink 2021)). For each class, we prepared at least 500 real photos of the brick and 1500 renders. Each such set was divided into training, test and evaluation set according to the following scheme. For renders, 90% of images were used as a training set and 10% were used as a validation set. For real photos, 60% of images were used in the training set, 20% for the validation set, and the remaining 20% as the test set. To find the best model for bricks labeling we tested the following three:

- InceptionV3 (for transfer learning with an added layer with 1024 neurons),
- VGG16 (for transfer learning two last layers were retrained),
- Xception (for transfer learning two last layers were retrained).

All the aforementioned architectures are widely used across the industry. Each model was trained using transfer learning. Transfer learning allowed us to achieve very good results with relatively small training sets and short training times. As the base model for transfer learning, we selected the one trained on the ImageNet set. In all further tests, we also trained the InceptionV3 model without transfer learning for reference.

As we can see in Fig. 6 all models trained using the transfer learning approach achieved very good results on the validation set even after the first epoch – the precision ranged from 0.685 to 0.81.

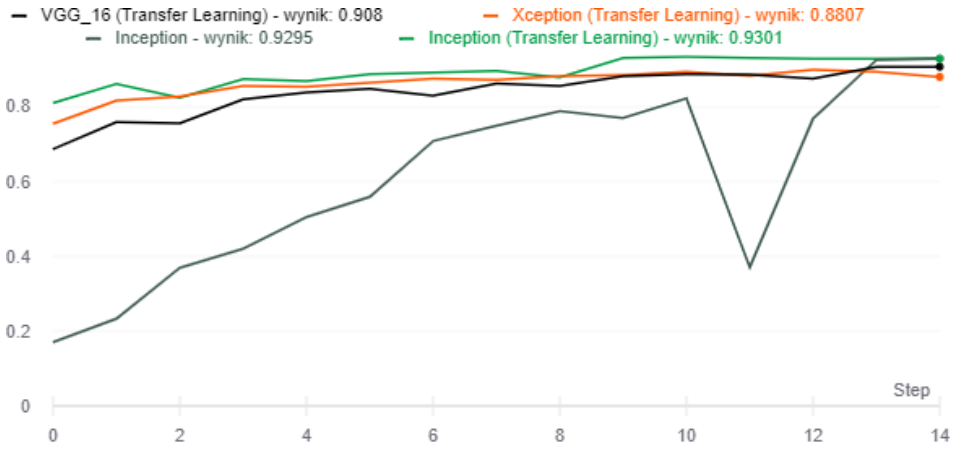

Fig. 6: Precision of bricks labeling.

As we can see the best results are achieved with the InceptionV3 model trained using the transfer learning approach for 9 epochs. The same model trained without transfer learning required 14 epochs to achieve similar results. Furthermore, this model showed a considerable precision drop-in epoch 11.

Finally, we decided to use the InceptionV3 model as it proved to achieve the best results. It also required the lowest number of epochs of training. Thanks to the separation of the detection and labeling steps we managed to obtain a valid solution able to detect LEGO bricks even in more complicated situations where other objects are present. Sample output of our test application can be seen in Fig. 7.

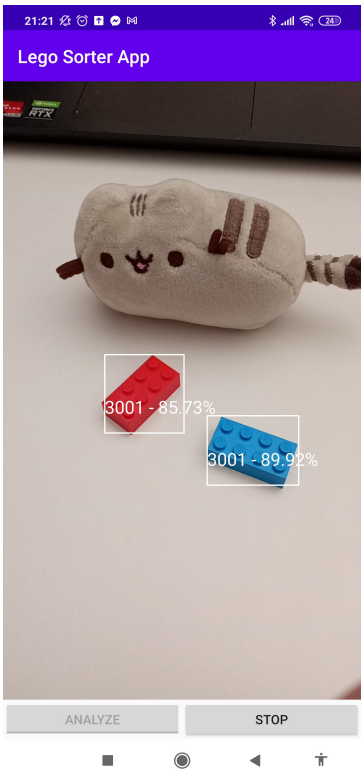

Fig. 7: Sample output of LEGO detection and labeling app.

Summary and Future Work

The proposed approach allows the detection of LEGO bricks and their labeling. The 2-step process proved to provide good results. Both the detection and annotation stages perform very well reaching over 90% of accuracy.

Currently, the solution allows the labeling of 10 different LEGO bricks. In our future works, we plan on extending the network with the ability to distinguish other types of bricks. We hope, that due to the usage of well-established models, a viable solution able to label more different LEGO bricks can be established. This will allow the creation of an AI-controlled LEGO sorting machine, which is the ultimate goal of the current research.

As a byproduct of the research, we also managed to establish a quick method of data generation. Both the detection and labeling models can work with multiple objects at once. This way we can quickly generate new training data by simply detecting multiple objects at once in the camera view. Providing that all the bricks are of the same type we can thus quickly create annotated data-sets used to train the labeling network.

Acknowledgements

I would like to thank my students, Konrad Zawora, Sławomir Zaraziński, Bartosz Śledź and Bogusław Łobacz for their invaluable help in the implementation of the presented solution.

References

- Adam (2019). “LEGO Sorting chart.” [Online], [Retrieved February 08, 2021], https://go.gliffy.com/go/publish/12232322.
- Aplhin, Tom (2020). “The LEGO Storage Guide.” [Online], [Retrieved February 08, 2021], https://brickarchitect.com/guide/.
- Bricklink (2021). “Parts Catalog.” [Online], [Retrieved February 08, 2021], https://www.bricklink.com/catalogTree.asp?itemType=P.

- Erhan, Dumitru et al. (2014). “Scalable object detection using deep neural networks.” In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 2147-2154.
- Everingham, M. et al. (2012). “The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results.” [Online], [Retrieved February 08, 2021], http://host.robots.ox.ac.uk/pascal/VOC/voc2012/results/index.html.
- Galvez, Reagan L et al. (2018). “Object detection using convolutional neural networks.” In: TENCON 2018-2018 IEEE Region 10 Conference. IEEE, pp. 2023-2027.
- Girshick, Ross (2015). “Fast R-CNN.” arXiv: 1504.08083 [cs.CV].
- Girshick, Ross et al. (2014). “Rich feature hierarchies for accurate object detection and semantic segmentation.” In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 580-587.
- Jocher, Glenn et al. (Jan. 2021). “ultralytics/yolov5: v4.0 - nn.SiLU() activations, Weights & Biases logging, PyTorch Hub integration.” In: doi: 10.5281/zenodo.4418161.
- Lin, Tsung-Yi et al. (2018). “Focal Loss for Dense Object Detection.” arXiv: 1708.02002 [cs.CV].
- Liu, Wei et al. (2016). “SSD: Single Shot MultiBox Detector.” In: Lecture Notes in Computer Science, pp. 21-37. issn: 1611-3349. doi: 10.1007/978- 3-319-46448-0_2.
- Maren, Tove (2018). “60 Fun LEGO Facts Every LEGO Fan Needs to Know.” [Online], [Retrieved February 08, 2021], https://mamainthenow.com/fun-lego-facts/.
- Pan, Sinno Jialin and Qiang Yang (2009). “A survey on transfer learning.” In: IEEE Transactions on knowledge and data engineering 22.10, pp. 1345-1359.
- Redmon, Joseph et al. (2016). “You Only Look Once: Unified, Real-Time Object Detection.” arXiv: 1506.02640 [cs.CV].
- Russakovsky, Olga et al. (2015). “Imagenet large scale visual recognition challenge.” In: International journal of computer vision 115.3, pp. 211-252.
- Simonyan, Karen and Andrew Zisserman (2014). “Very deep convolutional networks for large-scale image recognition.” In: arXiv preprint arXiv:1409.1556.
- Szegedy, Christian, Wei Liu, et al. (2014). “Going Deeper with Convolutions.” arXiv: 1409.4842 [cs.CV].
- Szegedy, Christian, Alexander Toshev, and Dumitru Erhan (2013). “Deep neural networks for object detection.”
- Tan, Mingxing, Ruoming Pang, and Quoc V Le (2020). EfficientDet: “Scalable and efficient object detection.” In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 10781-10790.
- Tensorflow (2020). “TensorFlow 2 Detection Model Zoo.” [Online], [Retrieved February 08, 2021], https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_detection_zoo.md.
- Torrey, Lisa and Jude Shavlik (2010). “Transfer learning. In: Handbook of research on machine learning applications and trends: algorithms, methods, and techniques.” IGI global, pp. 242-264.
- Tzutalin (2015). “LabelImg.” [Online], [Retrieved February 08, 2021], https://github.com/tzutalin/labelImg.
- ultralytics (2021). “YOLO v5 models.” [Online], [Retrieved February 08, 2021], https://github.com/ultralytics/yolov5/tree/master/models, files yolov5s.yaml and yolov5m.yaml.
- West, Daniel (2019). “LEGO sorting machine.” [Online], [Retrieved February 08, 2021], https://twitter.com/JustASquid/status/1201959889943154688.
- Yahalomi, Erez (2020). “Modular network for high accuracy object detection.” arXiv: 2001.09203 [cs.CV].
- Zawora, Konrad et al. (2021). “Tagged images with LEGO bricks.” Gdansk University of Technology. doi: 10.34808/2dbx-6a16.