Object Detection and Classification on Heterogeneous Datasets

Tobias Brosch and Ahmed Elshaarany
BMW Car IT GmbH, Lise-Meitner-Straße 14, Ulm, Germany

Keywords: Heterogeneous Datasets, Object Detection, Deep Learning, Faster R-CNN, Unlabeled Objects.

Abstract: To train an object detection network labeled data is required. More precisely, all objects to be detected must be labeled in the dataset. Here, we investigate how to train an object detection network from multiple heterogeneous datasets to avoid the cost and time intensive task of labeling. In each dataset only a subset of all objects must be labeled. Still, the network shall be able to learn to detect all of the desired objects from the combined datasets. In particular, if the network selects an unlabeled object during training, it should not consider it a negative sample and adapt its weights accordingly. Instead, it should ignore such detections in order to avoid a negative impact on the learning process. We propose a solution for two-stage object detectors like Faster R-CNN (which can probably also be applied to single-stage detectors). If the network detects a class of an unlabeled category in the current training sample it will omit it from the loss-calculation not only in the detection but also in the proposal stage. The results are demonstrated with a modified version of the Faster R-CNN network with Inception-ResNet-v2. We show that the model's average precision significantly exceeds the default object detection performance.

1 INTRODUCTION

Object detection and classification research has seen huge leaps over the past few years. Driven by the recent advances in object classification (Szegedy et al., 2014; Krizhevsky et al., 2012; Szegedy et al., 2016; Szegedy et al., 2015; Lin et al., 2014a; Zagoruyko and Komodakis, 2017; Xie et al., 2017) also object detection networks trained on annotated datasets were able to achieve very good results (Ren et al., 2016; Redmon et al., 2016; Redmon and Farhadi, 2018; Lin et al., 2018). To combat overfitting, models need to be trained with a large number of labeled images. Labeling, however, is a time and cost intensive task. Multiple approaches were introduced to augment datasets such as oversampling and image transformations. Still, the best results are achieved when models are trained on numerous instances of manually labeled images.

One solution is to take multiple datasets that contain at least labels for a subset of the required objects and to combine them. This, however, will lead to the following problem during training. Assume, for example, that we want to train a model to detect classes $C1, C2$, and $C3$ using two datasets. The first dataset, $DS1$, contains class labels for $C1$ and the second dataset, $DS2$, contains class labels for $C2$ and $C3$. Let's also assume that $DS1$ contains objects of $C3$ (which are not labeled). If the network correctly detects an object in an instance of $DS1$ of class $C3$ during training it will consider it as background (since it is not labeled in $DS1$) and will adapt its weights to not select it the next time (which of course has a negative impact on classification and detection performance).

In this work, we present an extension for two-stage object detectors that minimizes the described impacts. It can probably also be applied to single-stage detectors (left for future research). Performance is evaluated on an extended version of the the Faster R-CNN based network with Inception-ResNet-v2 and atrous convolutions pretrained on the COCO dataset (Lin et al., 2014b) provided by the TensorFlow object detection API (Huang et al., 2017) to train on multiple combined datasets. We show that the proposed model is significantly better than the original model when trained on hetereogeneous datasets.

2 RELATED WORK

Current state-of-the-art object detectors are either based on a two-stage proposal-driven mechanism or a one-stage detector. Through a sequence of advances the two-stage detectors (He et al., 2015; Girshick, 2015; Ren et al., 2016; Lin et al., 2017; He et al.,

Brosch, T. and Elshaarany, A.
Object Detection and Classification on Heterogeneous Datasets.
DOI: 10.5220/0007251903070312
In *Proceedings of the 8th International Conference on Pattern Recognition Applications and Methods (ICPRAM 2019)*, pages 307-312
ISBN: 978-989-758-351-3

2017) achieved top accuracy on the challenging COCO benchmark (Lin et al., 2014b). Same for single stage detectors like YOLO (Redmon et al., 2016; Redmon and Farhadi, 2018) and SSD (Liu et al., 2016) and recently out-performed two-stage detectors (Lin et al., 2018). For run-time comparisons see, for example, (Nguyen-Meidine et al., 2017).

None of those, however, had a focus on training from multiple datasets and on dealing with the problem of heterogeneous datasets, i.e., the combination of datasets in which each dataset contains potentially all objects but only a subset of all objects is labeled. In particular, a solution is needed to avoid punishing the network for correctly selecting an object that happens not to be labeled.

Note that this multi-task learning is somewhat related to inductive transfer learning. See (Pan and Qiang, 2010; Csurka, 2017) for comprehensive reviews on that topic. In contrast to inductive transfer learning, however, we simultaneously learn from the same source domains and multiple tasks (because of the differing label sets). In contrast to transfer learning, we are not only interested in the performance of the target domain but want to learn the target and source task simultaneously (besides, here, it cannot be clearly distinguished between source and target task).

It also has some relation to omitting the reward in reinforcement learning until a later time (Sutton and Barto, 1998; Mnih et al., 2015; Brosch et al., 2015; Brosch et al., 2013; Wörgötter and Porr, 2005; Grondman et al., 2012). The scenario here can be seen as not giving reward for a correctly detected instance.

3 PROPOSED MODEL

The proposed method is to combine the knowledge about what classes are not labeled in the particular training image with the network classification output during training. Whenever the network classifies an object as one that is not labeled in the current training sample it is omitted from the loss-calculation and consequently, does not harm the training process by giving erroneous feedback.

For single-stage detectors (Redmon et al., 2016; Redmon and Farhadi, 2018; Liu et al., 2016; Lin et al., 2018) we can simply omit training whenever the detector recognizes a class from which we know that it is not labeled in this particular training image. For two-stage detectors (Girshick, 2015; Ren et al., 2016; Lin et al., 2017; He et al., 2017) it is less obvious since the region-proposal stage is class agnostic and thus needs feedback from the classification stage. The first stage produces class agnostic region proposals (RP) and is trained based on the objectness and localization losses. The second stage on the other hand produces the bounding box detections and is trained based on classification and localization losses. Since the first stage produces RPs that are class agnostic, it is not possible to know which RPs are supposed to be ignored. Our suggestion to dealing with this problem is to wait for the second stage to classify the RPs produced by the first stage in an image. In this case, we know exactly which regions belong to the classes that are not labeled in that image based on which dataset it belongs to (c.f. Figure 1). Hence, any bounding boxes produced by the second stage and classified as a class that is not labeled in the current image would not contribute to the classification loss. Consequently, it will not affect the network weights.

The ignore-process for the first stage means that all region proposals that have an intersection over union of more than 0.5 do not contribute to the objectness and localization losses of the first stage (Figure 1, right, gray boxes). Similarly, the ignored bounding boxes of the second stage are not allowed to contribute to the classification and localization losses of the second stage. Thus, the loss is calculated like this:

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}} \sum_{i \notin \mathcal{N}_{cls}} L_{cls}(p_i, p_i^*) \quad (1)$$

$$+ \lambda \frac{1}{N_{reg}} \sum_{i \notin \mathcal{N}_{reg}} p_i^* L_{reg}(t_i, t_i^*) \quad (2)$$

In the example shown in Figure 1 only the green boxes would contribute to the loss-function (and the background anchors not shown) whereas the boxes shown in gray (that were classified as an object that is not being labeled in this training sample) are ignored. In contrast to (Ren et al., 2016), here, the loss is not summed over the set $\mathcal{N}_{cls,reg}$ of all indexes belonging to an area that was classified as an object that is not labeled for the particular training image. Otherwise, the loss is calculated as in (Ren et al., 2016). It consists of the classification loss $L_{cls}(p_i, p_i^*)$, i.e., the object vs. no-object loss between the ground-truth label p_i^* and the predicted probabilitiy p_i of anchor i being an object, and the regression loss $L_{reg}(t_i, t_i^*)$ that denotes the loss due to the difference between the predicted and the ground-truth bounding box (see (Ren et al., 2016) for further details). The training is then performed with stochastic gradient descent (LeCun et al., 1989) and the usual sampling strategies and hyperparameters as in (Ren et al., 2016) (see section 4 for details).

Here, we focus on two-stage detectors with an object vs. no-object stage and a separate classification stage. For single-stage detectors we expect similar results, which is to be investigated.

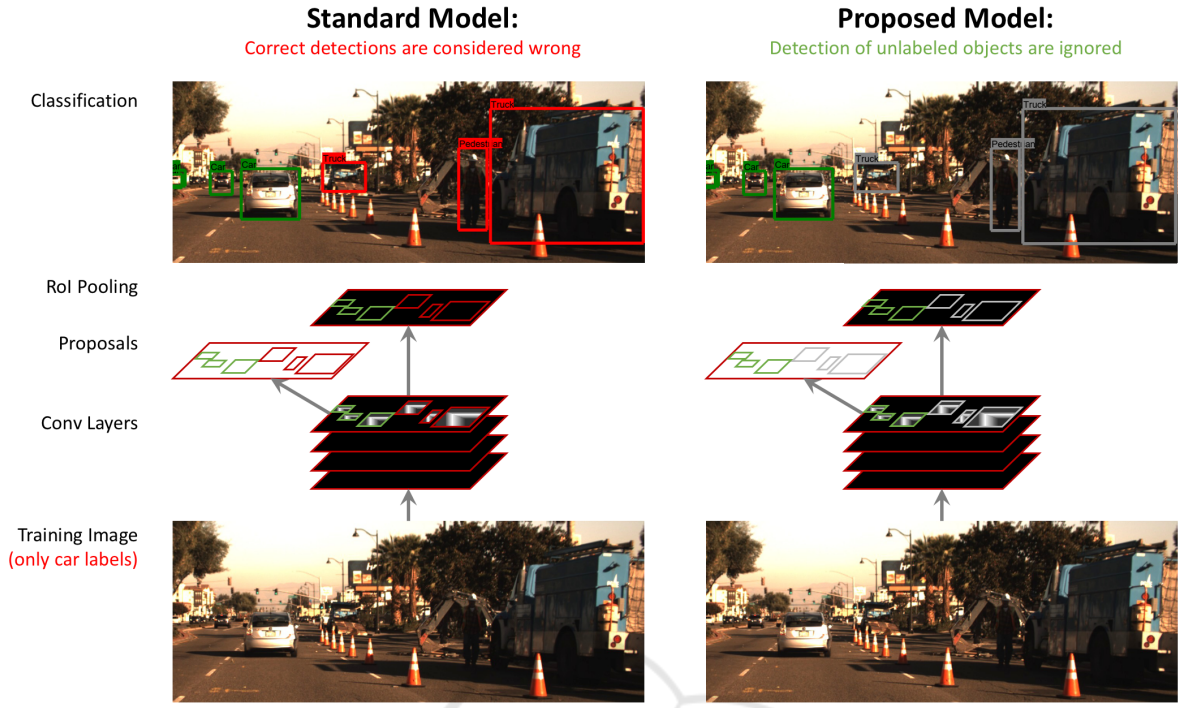

Figure 1: **The proposed model omits unlabeled objects to improve the training process:** In this example only cars are labeled in the input image. **Left:** In the standard Faster-R-CNN model all detections will contribute to the loss including the detections of trucks and pedestrians (shown in red). Since those objects are not labeled they will be considered as false positives and affect the network weights. **Right:** In the proposed model, however, the model knows that trucks and pedestrians are not labeled and consequently ignores all detections of such objects in the loss calculation of the "RoI" and "Proposal" stage (boxes shown in gray). Note: For illustration purposes no anchor boxes of the background are shown. Pictures from (Udacity, 2017).

4 RESULTS

In this section, we demonstrate that the proposed model-loss extension outperforms the original implementation if trained from heterogeneous datasets by a significant margin. In the following, we will explain the implementation (sect. 4.1), the employed datasets (sect. 4.2), the test configuration setup (sect. 4.3), and the test results (sect. 4.4).

4.1 Implementation

The proposed method was benchmarked with the TensorFlow Object Detection API[1]. More precisely, we extended the Faster R-CNN based network with Inception-ResNet-v2 and atrous convolutions (Szegedy et al., 2016) pretrained on the COCO dataset (Lin et al., 2014b) with our proposed loss-calculation method. For evaluation, we assembled train- and test-sets based on the Udacity datasets 1 and 2 (Udacity, 2017) (see next section for details).

[1]https://github.com/tensorflow/models/tree/master/research/object_detection

4.2 Datasets

In order to train the proposed method, we used Udacity dataset 1, created one subset with cars only (*DS*1), and one with trucks and pedestrians only (*DS*2). Both sets were then used to train the model. Mean average precision is reported for evaluation on Udacity dataset 2 (*DStest*).

As mentioned earlier, the problem with training on heterogeneous datasets is that not all classes are labeled in each and every dataset. In our test case, for example, *DS*1 has images with labeled cars (∼32000 instances), and *DS*2 has labeled trucks (∼2100 instances) and pedestrians (∼2100 instances) only (c.f. Figure 2, top for an example of *DS*1, and bottom for an example of *DS*2). During training on the bottom image of Figure 2, the original model would treat the unlabeled cars as negative examples because they do not have ground truth references (c.f. Figure 1, left). The same can occur for images that contain unlabeled pedestrians and trucks in *DS*1.

Figure 2: **Illustration of used datasets: Top:** Sample image from ***DS1*** with labeled cars. **Bottom:** Sample image from ***DS2*** with labeled pedestrians/trucks and unlabeled cars. The samples are taken from Udacity's annotated driving dataset by CrowdAI (Udacity, 2017).

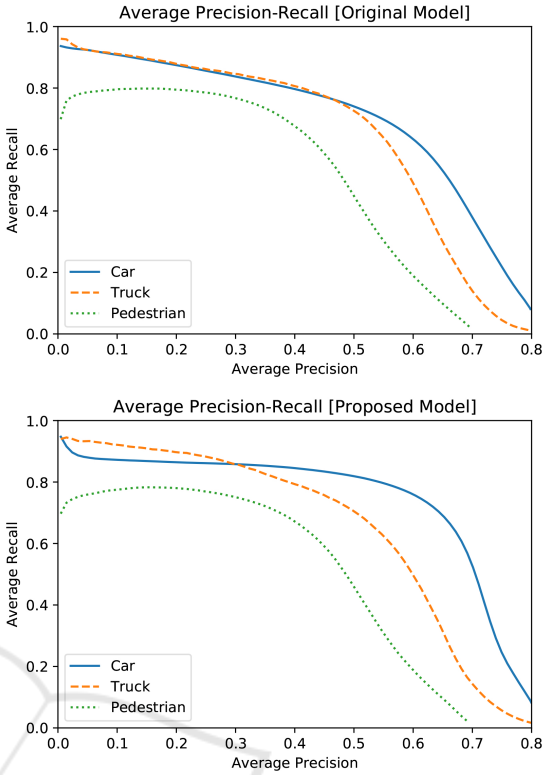

Figure 3: **Average precision-recall curves for each class. Top:** Original model. **Bottom:** Proposed model. Note that in particular the class of *cars* benefits significantly from the proposed model, which is to be expected due to a huge amount of samples of cars in the dataset. The curves shown are the average of 30 runs.

4.3 Test Configuration

We benchmarked the proposed method against the unmodified Faster R-CNN version of the TensorFlow Object Detection API. Sampling strategies and hyperparameters were identical for both models. Only the loss calculation differed for the proposed model as outlined in sect. 3.

4.4 Test Results

Overall the mean average precision was 54.4% for the modified variant which was significantly better than the mean average precision of 53.2% of the original model (on 1% confidence-level, $n = 30$ test-runs). The mean average precisions for trucks and pedestrians were almost identical whereas the mean average precisions for cars were significantly better for the modified variant (64.6% vs. 60.9% for the modified vs. the original model on 1% confidence-level, $n = 30$ test-runs), which is to be expected due to the huge number of cars in the dataset, since the original model suffers significantly from being "punished" to correctly select cars.

A comparison of the average precision-recall curves between the original and the proposed model confirms those observations. They show that in particular the class of *cars* benefits from our proposed model (Figure 3).

Note that this also demonstrates that the original model is already quite robust with respect to unlabeled objects in the dataset as long as the objects are not too frequent or do not occupy too much of the scene (because in this case the likelihood of being selected as training sample is higher). Thus, we expect that our model performs even better compared to the original model for objects that take up a lot of space on images and/or are very frequent. This is definitely something that needs to be investigated in more detail in the future.

Finally, in order to assess whether additional labels would lead to a better classification result, we also used *DS*1 with no objects being unlabeled (i.e. the original dataset). Most interestingly, the original model trained on this complete and fully labeled dataset did not outperform the proposed model, demonstrating that the proposed method in this case achieves the same performance level without the need for a fully labeled training set.

5 CONCLUSION

We proposed a novel method to train a two-stage object detection network from multiple datasets in which each dataset does not need to have the full label set, i.e. not all object categories are labeled in all datasets that are used for training. The results indicate that the novel approach outperforms a regular object detection network significantly by excluding unlabeled objects from the loss-calculation. Furthermore, the results indicate that depending on the task even regular approaches are quite robust but can perform better when extended with the new method which excludes regions from the loss-calculations that have been identified as objects of an unlabeled category for the current training sample. Thus, our method can help to speed up learning of new object sets without going through the time and cost intensive task of labeling all objects in the entire dataset. It also helps in domains where labeled data is rare. From a run-time perspective the proposed method is virtually identical to the original Faster R-CNN implementation.

In addition to the study presented here, more work is needed. In future studies, additional dataset configurations need to be evaluated and it also needs to be investigated how the method performs with single-stage detectors like (Redmon et al., 2016; Redmon and Farhadi, 2018; Liu et al., 2016; Lin et al., 2018). It might also be interesting to see if the approach can be transferred to other domains such as action recognition (Layher et al., 2017). Furthermore, it should also be adressed how many datasets can be simultaneously used and how it affects system performance.

ACKNOWLEDGEMENTS

We thank Philippe Chiberre for his work on preliminary versions of the ideas outlined in this paper.

REFERENCES

Brosch, T., Neumann, H., and Roelfsema, P. R. (2015). Reinforcement Learning of Linking and Tracing Contours in Recurrent Neural Networks. *PLoS Computational Biology*, 11(10):e1004489.

Brosch, T., Schwenker, F., and Neumann, H. (2013). Attention–Gated Reinforcement Learning in Neural Networks–A Unified View. In *ICANN*, volume 8131 of *LNCS*, pages 272–9. Springer.

Csurka, G. (2017). Domain Adaptation for Visual Applications: A Comprehensive Survey. In G., C., editor, *Domain Adaptation in Computer Vision Applications*, chapter Advances in Computer Vision and Pattern Recognition, pages 1–35. Springer.

Girshick, R. (2015). Fast R–CNN. In *Proceedings of the 2015 IEEE International Conference on Computer Vision*, ICCV, pages 1440–8. IEEE.

Grondman, I., Buşoniu, L., Lopes, G. A. D., and Babuška, R. (2012). A Survey of Actor–Critic Reinforcement Learning: Standard and Natural Policy Gradients. *Systems, Man, and Cybernetics*, 42(6):1291–1307.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017). Mask R–CNN. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2980–8. IEEE.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1904–16.

Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., Fischer, I., Wojna, Z., Song, Y., Guadarrama, S., and Murphy, K. (2017). Speed/Accuracy Trade–Offs for Modern Convolutional Object Detectors. https://arxiv.org/pdf/1611.10012.pdf.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In *NIPS*.

Layher, G., Brosch, T., and Neumann, H. (2017). Real–Time Biologically Inspired Action Recognition from Key Poses Using a Neuromorphic Architecture. *Frontiers in Neurorobotics*, 11(13):1–21.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation*, 1(4):541–51.

Lin, M., Chen, Q., and Yan, S. (2014a). Network in Network. https://arxiv.org/pdf/1312.4400v3.pdf.

Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., and Belongie, S. (2017). Feature Pyramid Networks for Object Detection. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 936–44. IEEE.

Lin, T.-Y., Goyal, P., Grishick, R., He, K., and Dollár, P. (2018). Focal Loss for Dense Object Detection. https://arxiv.org/pdf/1708.02002.pdf.

Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C. L., and Dollár, P. (2014b). Microsoft COCO: Common Objects in Context. In *Computer Vision – ECCV 2014*, pages 740–55. Springer.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. (2016). SSD Single Shot MultiBox Detector. https://arxiv.org/abs/1512.02325.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C.and Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. (2015). Human–Level Control Through Deep Reinforcement Learning. *Nature*, 518:529–33.

Nguyen-Meidine, L. T., Granger, E., Kiran, M., and Blais-Morin, L.-A. (2017). A Comparison of CNN–based Face and Head Detectors for Real–Time Video Surveillance Applications. In *Seventh International Confer-*

ence on Image Processing Theory, Tools and Applications (IPTA), pages 1–8. IEEE.

Pan, S. J. and Qiang, Y. (2010). A Survey on Transfer Learning. *IEEE Trans. on Knowl. and Data Eng.*, 22(10):1345–1359.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You Only Look Once: Unified, Real–Time Object Detection. https://arxiv.org/pdf/1506.02640.pdf.

Redmon, J. and Farhadi, A. (2018). YOLOv3: An Incremental Improvement. https://arxiv.org/pdf/1804.02767.pdf.

Ren, S., He, K., Girshick, R., and Sun, J. (2016). Faster R–CNN: Towards Real–Time Object Detection with Region Proposal Networks. 39(6):1137–49.

Sutton, R. S. and Barto, A. G. (1998). *Reinforcement Learning: An Introduction*. MIT Press, London, England.

Szegedy, C., Ioffe, S., Vanhoucke, V., and Alemi, A. (2016). Inception–v4, Inception–ResNet and the Impact of Residual Connections on Learning. https://arxiv.org/pdf/1602.07261.pdf.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2014). Going Deeper with Convolutions. http://arxiv.org/abs/1409.4842.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2015). Rethinking the Inception Architecture for Computer Vision. https://arxiv.org/pdf/1512.00567.pdf.

Udacity (2017). Udacity Self Driving Car Dataset. https://github.com/udacity/self--driving--car/tree/master/annotations.

Wörgötter, F. and Porr, B. (2005). Temporal Sequence Learning, Prediction, and Control: A Review of Different Models and Their Relation to Biological Mechanisms. *Neural Computation*, 17(2):245–319.

Xie, S., Girshick, R., Dollar, P., Tu, Z., and He, K. (2017). Aggregated Residual Transformations for Deep Neural Networks. https://arxiv.org/pdf/1611.05431.pdf.

Zagoruyko, S. and Komodakis, N. (2017). Wide Residual Networks. https://arxiv.org/pdf/1605.07146.pdf.