LEARNING WITH REAL-WORLD AND ARTIFICIAL DATA FOR IMPROVED VEHICLE DETECTION IN AERIAL IMAGERY

I. Weber[1,*], J. Bongartz[1], R. Roscher[2],

[1] University of Applied Sciences Koblenz, AMLS, Remagen, Germany - (immanuel.weber, bongartz)@hs-koblenz.de
[2] Institute of Geodesy and Geoinformation, University of Bonn, Germany - ribana.roscher@uni-bonn.de

Commission II

KEY WORDS: Object detection, deep learning, data generation, multi-source learning

ABSTRACT:

Detecting objects in aerial images is an important task in different environmental and infrastructure-related applications. Deep learning object detectors like RetinaNet offer decent detection performance; however, they require a large amount of annotated training data. It is well known that the collection of annotated data is a time consuming and tedious task, which often cannot be performed sufficiently well for remote sensing tasks since the required data must cover a wide variety of scenes and objects. In this paper, we analyze the performance of such a network given a limited amount of training data and address the research question of whether artificially generated training data can be used to overcome the challenge of real-world data sets with a small amount of training data. For our experiments, we use the ISPRS 2D Semantic Labeling Contest Potsdam data set for vehicle detection, where we derive object-bounding boxes of vehicles suitable for our task. We generate artificial data based on vehicle blueprints and show that networks trained only on generated data may have a lower performance, but are still able to detect most of the vehicles found in the real data set. Moreover, we show that adding generated data to real-world data sets with a limited amount of training data, the performance can be increased significantly, and in some cases, almost reach baseline performance levels.

1. INTRODUCTION

Object detection in aerial images is an important task in remote sensing applications like environmental monitoring, infrastructure surveillance, or traffic monitoring (Heipke, Rottensteiner, 2020, Ma et al., 2019). Deep neural networks like RetinaNet (Lin et al., 2017b), which are specifically designed for object detection, have shown to be suitable tools for solving such a task (Lin et al., 2017b, Sun et al., 2018). They have proven their capabilities in different benchmarks covering general imagery but also aerial imagery (Lin et al., 2014b, Xia et al., 2018). However, they share the same problem as most deep learning methods in that they require a vast amount of annotated training data. On the contrary, in real-world applications, the lack of training data is often imminent. This is especially true when well generalizing models are required to be applicable over a wide range of situations varying in the scenery, structures, weather, and lighting conditions, and the variants of objects to detect are even larger. Many times collecting suitable training data covering this wide span is not possible or prohibitively expensive. Therefore, much research has been dedicated to reduce the necessary amount of data to train a model to reach adequate performance and generalizability.

Several approaches have been proposed to overcome this challenge. The most common, yet limited one, is data augmentation, where the amount of training data is increased by geometric and spectral transformations. However, this approach uses information from training data only, which may result in a possibly not representative training set. More sophisticated methods generate artificial data from 3D models or use generative adversarial networks (see Section 2), but they lead to a high degree of complexity in the generation pipeline or require large amounts of computing power.

*Corresponding author

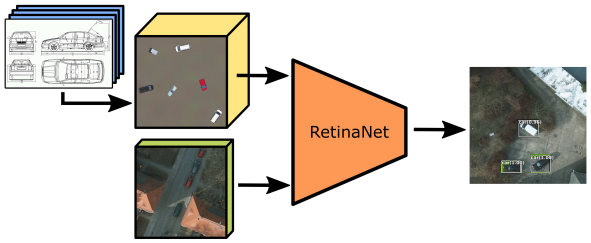


Figure 1. Combining artificial imagery, generated based on vehicle blueprints, with small real-world data can significantly improve the performance of an object detection network like RetinaNet.

Our requirements for an artificial training data generation pipeline, however, are to keep the generation process relatively simple and usable for diverse applications. Therefore, comprehensive 3D modeling, texturing, and generation of a multitude of possible object-background combinations should be avoided. Ultimately, we aim for the disentanglement of relevant objects and backgrounds. To this end, we consider image augmentation and similar approaches as complementary steps to further increase variability.

Therefore, we focus on a simple generation process and access its impact on a deep neural network. We base our analyses on the task of vehicle detection in imagery from the ISPRS 2D Semantic Labeling Contest Potsdam dataset (Rottensteiner et al., 2013). Our contributions are as follows:

- We first evaluate the impact of limited amounts of real-world training data.
- Building on that we test what detection performance can

be achieved with training data consisting of purely artificial data, and how the generation pipeline can be modified to improve the detection performance.

- Finally, we evaluate training models with joined real-world and artificial data (Fig. 1).

The paper is structured as follows: The process of deriving object bounding boxes from the ISPRS 2D Semantic Labeling Contest Potsdam and creating a dataset from this is described in Section 3. Section 4 presents the mentioned experiments, and Section 5 closes this work with conclusions and a small outlook.

2. RELATED WORK

Different methods have been proposed which increase the amount of training data and its variability, either by diversifying the original training data or by generating new training data. Simple geometric and radiometric image augmentation methods, like flipping or color changes, are well known to increase the data variability, are easy to integrate, and can increase a model's performance significantly (Liu et al., 2016). More advanced methods like Cutout (DeVries, Taylor, 2017), Mixup (Zhang et al., 2017) or CutMix (Yun et al., 2019) try to broaden the networks receptive field and remove strong location dependencies. There are also specific augmentations for object detection (Zhang et al., 2019); some of them are, however, strongly dependent on the network structure (Yun et al., 2019).

Semi-supervised learning methods try to circumvent the need for large scale annotated data sets by designing smart but simple so-called pretext tasks, which can work on non-annotated data sets (Berthelot et al., 2019, Nair et al., 2019). Both pretext tasks and main tasks share central parts of the network architecture, hence learning of the former also improves the latter. In principle, semi-supervised learning and image augmentations are tightly related, as data for pretext tasks is often generated by augmentation methods. Both approaches have in common that they can not create new data.

A more far-reaching approach is to generate artificial data by 3D rendering systems (Movshovitz-Attias et al., 2016) or 3D video game engines (Richter et al., 2016). Generally, these approaches can generate quite sophisticated artificial imagery for many applications. (Peng et al., 2015, Tremblay et al., 2018), for example, show that generating imagery from 3D models can be on par with networks trained on pure real-world data. However, a reasonable quality of the modeled scenes and objects can be laborious or challenging to achieve. Compared to these works, aerial imagery can be generated more easily, as the scenery is typically only viewed in a top-down manner; hence only the top view of the objects has to be generated. Also, generative adversarial networks (GAN) can be used to generate artificial data, which serves as additional training data (Zheng et al., 2019). However, these networks are generally complex and remain relatively challenging to optimize.

Another approach is the context-aware combination of real and artificial data. By using semantic maps, objects such as vehicles could be inserted on semantically appropriate surfaces such as roads only. However, this requires additional data, and still results in the generation of a large amount of data with a relatively complex generation process.

3. DATA GENERATION

We base our work on the ISPRS 2D Semantic Labeling Contest Potsdam data set (in the following abbreviated as Potsdam data set) [1], which consists of 38 image patches covering part of the city of Potsdam (Rottensteiner et al., 2013). We are aware that other data sets, like DOTA (Xia et al., 2018) and COWC (Mundhenk et al., 2016), exist that are specifically designed for object detection. However, we aim for a higher ground resolution than the imagery in DOTA, and COWC vehicles are annotated with their center points only instead of bounding boxes, which is not suitable for us. Also, the imagery of the Potsdam data set matches our target domain the most. The Potsdam data set patches are part of a true orthophoto and have a ground sampling distance of $5\,\mathrm{cm}$ and a size of 6,000 $\times$ 6,000 px. As the data set is part of a semantic labeling benchmark, semantic annotations are provided accordingly. The annotations contain data for five classes (*impervious surfaces*, *building*, *low vegetation*, *tree*, *car*, *clutter background*). These are realized using semantic maps with a distinct color for each of the classes. The data set is split into 24 patches for training and 14 for validation.

3.1 Deriving object detection annotations

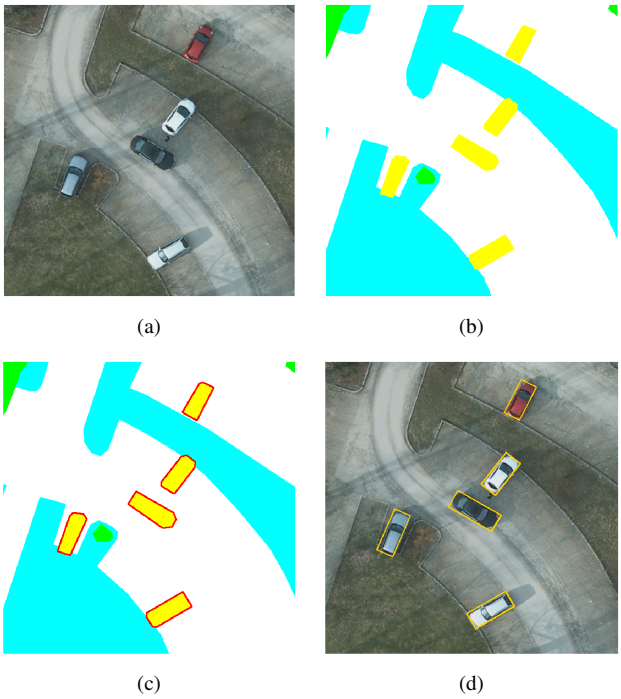
The Potsdam data set is designed for semantic annotation and is not readily usable for object detection. Before we transform its annotation masks to bounding box annotations, we first cut the image patches into smaller subpatches. This is necessary as object detection networks usually do not work directly on images of a size that large as it would computationally be too expensive. We slice them into subpatches of size 600 $\times$ 600 px in a sliding window fashion, with an overlap of 200 px in both directions, which corresponds to $10\,\mathrm{m}$. This should allow for most objects annotated as *car* to be at least fully visible in one subpatch. The subpatches are padded to the target size if necessary, and the padding color is chosen to be the mean color of the ImageNet data set (Deng et al., 2009), which after image normalization in the preprocessing will result as values of 0. This slicing results in 5400 training and 3150 validation subpatches.

For the transformation of the annotations (Fig.2), we concentrate on the class *car*, as vehicle detection is our intended task, and extract only the corresponding color from the annotation subpatches. We ignore all other classes. Then we determine the contours of the contained objects and afterward compute the corresponding smallest rectangles enclosed in the contours. Based on those, the axis-aligned bounding boxes can be computed. One could argue that the bounding boxes can be computed directly on the contours; however, in some situations, we found that the former approach creates better boxes, for example, in the case of partially concealed cars. These boxes are then exported in the COCO (Lin et al., 2014b) object annotation format[2].

As some vehicles are largely concealed by buildings or vegetation or are only partially visible due to the slicing, we remove annotations for whose both sides are smaller than $1\,\mathrm{m}$, and a single side is smaller than $2\,\mathrm{m}$. Some larger vehicles like trucks, which are part of the data set and are relatively rare, would likely distort the data distribution. Hence we limit annotations on the largest side to be below $10\,\mathrm{m}$. Also, we manually remove the annotation of a single tram and separate some

[1] http://www2.isprs.org/commissions/comm3/wg4/2d-sem-label-potsdam.html

[2] http://cocodata set.org/#format-data



(a) (b) (c) (d)

Figure 2. Transformation of the semantic annotation to bounding boxes: (a) and (b) show the RGB and annotation subpatches; the yellow color corresponds to the class *car* which is used in (c) to create the contours (shown in orange) for each object belonging to that class; (d) shows the derived minimal area rectangle for each object overlayed on thr RGB subpatch, from these the bounding boxes can be computed.

touching annotations, which otherwise result in single bounding boxes covering multiple vehicles. Besides that, we remove subpatches that do not contain any vehicle. Our final data set consists of 2824 and 1828 subpatches remaining for training and validation, containing 12182 and 7924 objects of the class *car*.

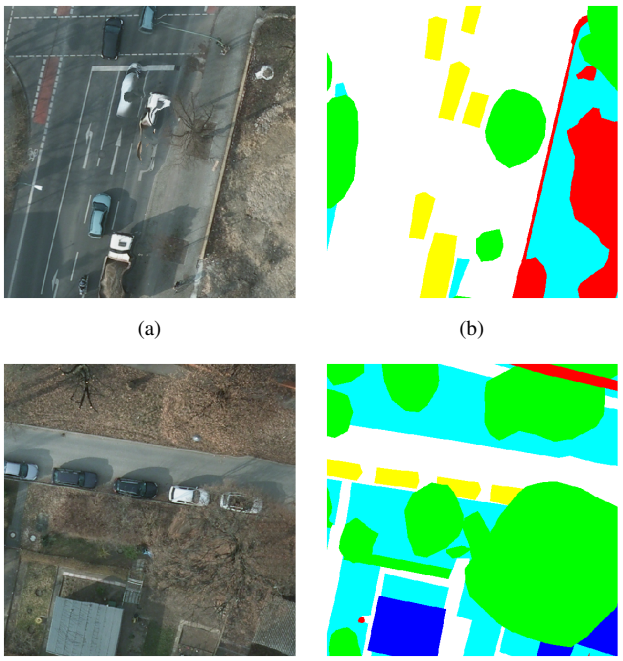
3.2 Deficiencies of the derived data set

Some areas of the image patches contain artifacts (Fig. 3a, 3b), which stem from the structure from motion process, which was used to construct the true orthophoto. Another issue (Fig. 3c, 3d) is that the imagery of the data set was captured in late autumn or wintertime, where most of the trees were leafless. The data set is annotated in a way that the topmost class determines the annotation; hence cars parked underneath those trees are not or only partly annotated, even though they are visible to the human observer. This is a challenge for the object detection network, as we will show later in our experiments, and will affect the detection performance.

3.3 Generation of artificial data

We generate artificial vehicle imagery based on manufacturers' blueprints. For this, we select eight typical vehicles ranging from small cars, over station wagons, SUVs, and vans (Fig. 4).[3]

We remove elements that are not part of the car, such as size markings. Then we create masks using different colors for background, vehicle body, and windows (Fig. 5b) and rescale



(a) (b) (c) (d)

Figure 3. Examples of data set issues: (a) and (b) show a RGB subpatch with fragments and the corresponding annotation; (c) and (d) show a RGB subpatch with cars partially or fully covered by a leafless tree and corresponding annotation subpatch, which has only annotations for the not covered car parts.

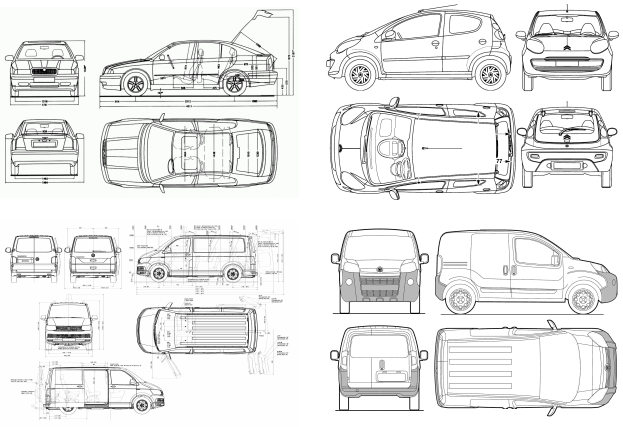


Figure 4. Examples of car blueprints used to generated the artificial data.

[3] https://drawingdatabase.com/

them to match our targeted ground sampling distance. We randomly color the vehicle body with colors typical for cars and the windows with blueish colors (Fig. 5c). Both colors are slightly randomly altered to increase variance. Per image we create four of these vehicles and, also two partial vehicles, as many of the vehicles in the real-world data set are only partially visible. The background is covered in the mean color of the ImageNet data set and has a combination of a very coarse and a fine spatial noise pattern added to it. We evaluate the influence of this process in the experiments.

4. EXPERIMENTS

Our object detection network is a customized RetinaNet (Lin et al., 2017b) structure, with a Resnet-50 (He et al., 2016) feature extractor back end. The feature extractor is combined with a feature pyramid network (Lin et al., 2017a), which is the base of the front end and consists of five layers, as published in (Lin et al., 2017b). Deeper extractors do not perform significantly better in our case; hence we use the 50-layer version to keep training and inference times low. The detections are computed using the feature-layer-wise shared subnets for classification and localization. The detection front end is designed to process images of size 300×300 px, hence the total amount of anchors is 17451, consisting of anchor areas from 16×16 to 256×256. The anchor areas differ from the ones described in (Lin et al., 2017b) by a factor of four, because these were designed for maximum input sizes of 800×1333 and would be too large for our input sizes. The classification loss function is the accompanying FocalLoss, which strictly requires the correct initialization of the bias of the final layer of the classification subnet. Our regression loss is a smooth L1 loss.

We implement all our code using the pytorch (Paszke et al., 2019) framework in combination with the fast.ai (Howard et al., 2018) library. Our back end structure uses a model pretrained using the ImageNet data set (Deng et al., 2009), which is part of the torchvision (Paszke et al., 2019) model library. Each trial is computed on a single NVIDIA Tesla V100 GPU. All subpatches are rescaled from 600 to 300 px size during training and validation. Experimentation prior to this work has revealed that working directly with an input size of 600 px does not increase the detection performance significantly. Our image preprocessing during training augments the images using random flipping of both axes and random changes of brightness, contrast, hue, and saturation. For training and validation, the images are z-normalized using the mean and standard deviations of the ImageNet image colors. The batch size of the experiments is 32, if not specified otherwise. We train all our models using the SGD optimizer and the 1cycle learning rate policy as described by (Smith, 2018), which is readily available as a scheduler in the fast.ai library. The peak learning rate is 3×10^{-2} if not specified otherwise. Learning starts with $1/25$ of it and finishes with $1/25\,000$ of it[4]. The momentum is varied between 0.95 and 0.85. Both hyper-parameters peak at 30 % of the training schedule.

Over the full training process, we store the current best model and load it after the training has finished. This allows us to get access to the best model even if the model overfits and diverges. We also stop early if the validation loss does not decrease for more than 0.001 over ten epochs in a row. Our experimentation

[4]For a learning rate of 3×10^{-2} this means 2×10^{-3} and 2×10^{-7} respectively.

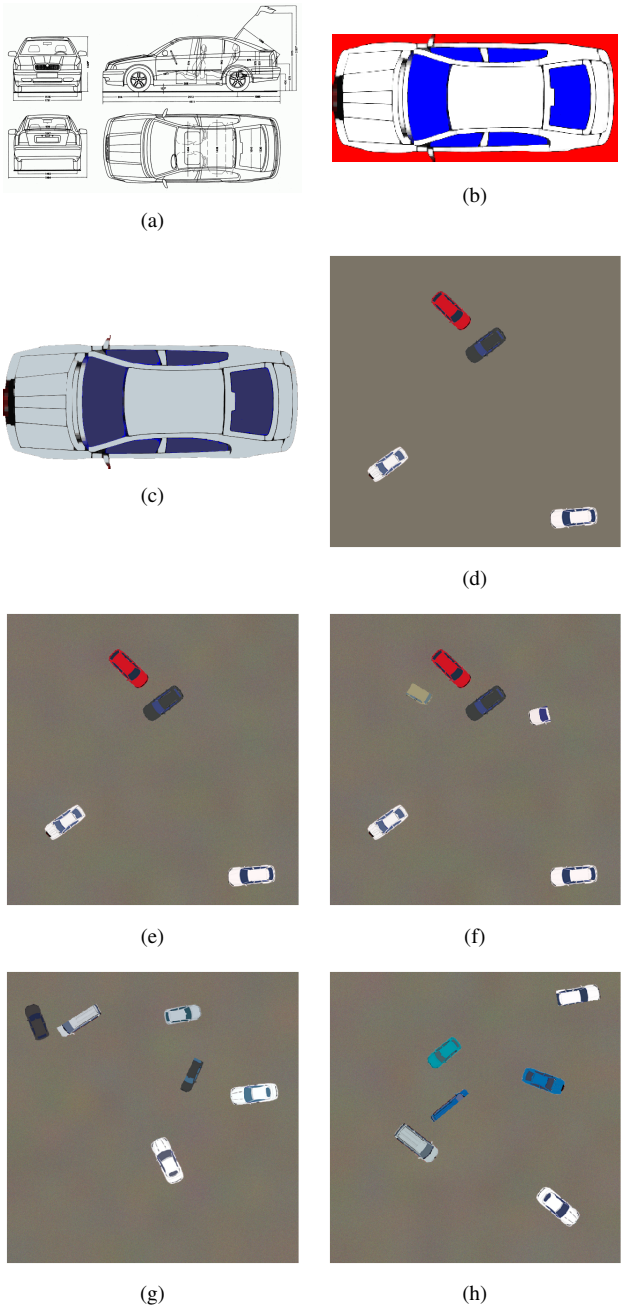


Figure 5. Artificial data generation: (a) the initial car blueprint; (b) cleared and color masked top view of the car; (c) randomly colored car; (d, e, f) show the same image width blank background, with noise added to the background and with vehicle parts; (g, h) show to more example images.

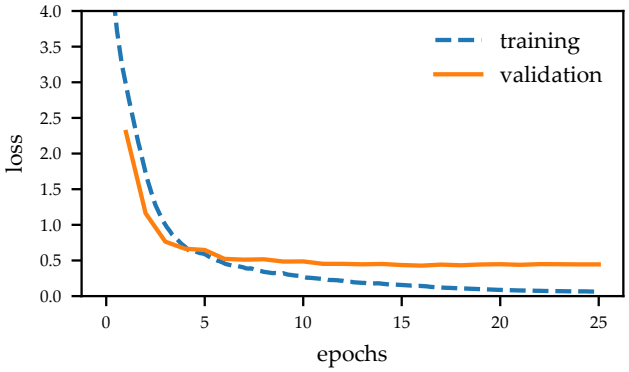


Figure 6. Training and validation loss for training the model with the complete real-world data set for 25 epochs, which takes about 10 minutes. The 1cylce training policy with a maximum learning rate of 3×10^{-2} is applied. The best losses at epoch 16 of 0.063 for training and 0.442 for validation result in 90.2 % AP.

has shown that using the 1cycle scheduler it is enough to train the model for 25 epochs to fully converge, which due to the relatively small amount of data, takes about 10 minutes.

For the evaluation of the object detection performance we use the average precision (AP) of the COCO data set (Lin et al., 2014a) [5]. We use an intersection over union threshold of 0.5, which matches COCO's $AP^{IoU=0.5}$. Our confidence threshold is 0.1 to keep a safe distance from the initial confidences of about 0.01, which result from the FocalLoss bias initialization. We do not apply any hard limit for the number of proposals, as after training, the number of valid proposals is generally quite limited.

4.1 Performance on complete and reduced real-world data sets

We begin our experimentation by establishing the baseline performance of our RetinaNet on the complete real-world training data set. For this, we train the model for 25 epochs and reach a baseline performance of AP = 90.2%. The AP is computed using the model state of epoch 16, where the lowest validation loss is reached. Figure 6 shows the training and validation loss curves for the training of the baseline model. In our experimentation, we found that the achievable average precision is most likely limited due to the characteristics of our derived data set. Fixing the described deficiencies may help to raise this limit.

We now determine the performance achievable of small data sets by reducing the size in steps of 1000, starting with 2500 samples and have finer decrements below 500 until we stop at eight samples. To keep the number of total iterations constant, we increase the number of epochs accordingly. In general, the model begins to overfit more rapidly as the data set size decreases. However, the training scheme still tends to result in a good model if we load the best model state after the training has finished. Table 1 shows the resulting average precision values, which keep steady over a broad range down to 500 samples, and only at about 50 samples largely drop.

4.2 Artificially generated data performance

In this section, we evaluate the object detection performance of our network trained on the generated artificial imagery. The

[5] http://cocodata set.org/#detection-eval

# training samples	AP [%]
2824	90.2
2500	90.1
1500	90.0
500	89.3
100	86.6
50	82.0
32	75.6
16	48.1
8	30.7

Table 1. Average precision for the complete and reduced real-world data sets used for training; we train the models for the data sets with 16 and 8 samples with a batch size of 16 and 8, respectively.

data is generated as described in Sec. 3.3. We evaluate three different variants to generate the data. Variant A only creates four vehicles on a blank background image, variant B modulates noise onto the blank background, and variant C adds two vehicle parts. We train a model for each of the variants with data sets consisting of 250, 500, 1000, and 1500 generated samples for 60 epochs using a peak learning rate of 1×10^{-2}.

Table 2 shows the average precision values for each variant and each of the data set sizes. The table shows that added noise to the background increases the performance significantly. We assume the reason for this is that noise likely helps the model to remove dependencies on a homogeneous background, which we do not observe in the real-world data, and so helps to generalize better. Adding vehicle parts also improves the performance as the real-world data set contains numerous partly visible objects. Based on this result, we choose to use 1000 artificial samples of variant C for the following experiment, as the improvement over 500 samples is significant, and more data does not result in much better performance.

4.3 Joining real-world and generated data

Using the insights gained in the previous section, we now combine both data sets. We take our best performing generated data set and add it to the reduced real-world data sets from section 4.1. Each model is trained for 60 epochs with a peak learning rate of 3×10^{-2}. The resulting average precision values are shown in Tab. 3 and are compared to the baseline in Fig. 7. It is visible that the generated data can lift the baseline performances of the small real-world data sets largely. There is a small negative impact on two of the larger data sets, but this can be neglected in practice. Moreover, a specific fine-tuning per data set can result in on par performance levels but is not shown here to keep hyper-parameters steady.

The samples in Fig. 8 show that the model trained solely on artificial data tends to make erroneous detections with relatively high confidence in image regions, which do not belong to the class *car*. We assume that the reason for this is that artificial images are generated with noise background rather than real-world background and thus, the network only learns to distinguish noise background from vehicles. Real-world background, however, may contain structures which are close to the vehicle distribution, that the model has learned, and hence may result in false positives. The detection samples from the models which are trained with additional 8 and 32 real-world images, however, clearly show the improvement gained from such few real images. The false positive detections are increasingly reduced, and the bounding boxes of the vehicles become more accurate.

Variant	AP_{250} [%]	AP_{500} [%]	AP_{1000} [%]	AP_{1500} [%]	AP_{2000} [%]
Variant A	38.9	51.2	47.7	53.4	55.5
+ Variant B: Background Noise	48.6	57.5	58.6	57.7	58.6
+ Variant C: Vehicle Parts	62.2	62.3	65.4	65.8	65.6

Table 2. The object detection performance of models only trained on generated data for the three variants. Each variant is computed over a range of data set sizes. For the following experiments, we use 1000 samples of variant C.

# real-world samples	AP [%]	Change [% points]
2824	90.2	
2500	90.2	+0.1
1500	89.8	-0.2
500	89.2	-0.1
100	86.7	+0.1
50	84.7	+2.7
32	83.4	+7.8
16	75.9	+27.8
8	73.3	+42.6
0	65.4	

Table 3. Average precision values and their changes in points of percentage compared to the baseline values for each joined data set consisting of the reduced real-world data set and 1000 generated images. We train each model for 60 epochs using a learning rate of 3×10^{-2}.

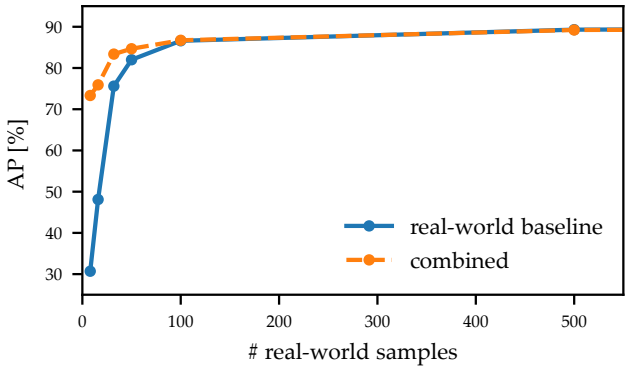


Figure 7. Average precision for models trained with an increasing amount of real-world data (solid blue line) and trained with a combination of the real-world data and 1000 generated images for comparison (orange dashed line). Please note that we only show the results up to 500 real-world samples, as from this point on, the average precision is almost constant.

4.4 Influence of data set deficiencies

The mentioned issues in the derived data set become visible during the inference of the trained network, and both are displayed in Fig. 9. The second detection from the top in the left image has a relatively high confidence; however, there is no annotation (no green box) as the vehicle is parked under a tree. This detection shows that the network can detect these vehicles even though it is not meant to learn those. These detections result in false positives, which decrease the precision, and in the end, harm the AP. The figure also shows that the network's detection of the lower car is better than the annotation. However, this does not benefit, but in fact, may worsen the AP value as the intersection over union between detection and annotation is used to assign proposals to annotations.

5. CONCLUSION AND FUTURE WORK

In this paper, we addressed vehicle detection in aerial imagery, for which only a limited amount of labeled samples is available. We analyzed the impact of learning with artificially generated data samples on the detection accuracy. Our experiments on real-world data set sizes show, that high object detection performance levels can be achieved for a wide range of sizes. It is worth to note that a reasonable high level can already be achieved with a few hundred samples and that larger data sets only slightly improve the performance. These observations may especially be true as we only consider one class, and the objects in our data set are relatively homogeneous, and hence a sufficient number of samples are available. However, we also see that at some point, the performance drops drastically, and the detection quality becomes insufficient.

We show that data sets consisting only of artificial data, which can easily be created from blueprints, can achieve higher performances than data sets with a limited amount of real-world training data. If we combine both, we can increase the limited performance levels largely and almost reach baseline levels.

In summary, we conclude that these observations point to research directions that are promising to efficiently reduce the need for large data sets. This is a relevant aspect, especially in remote sensing. However, more needs to be done to improve certain aspects. We need to get a better understanding of what is important for the network and what can be done to generate data that is relevant to achieve accurate and robust results. Here, the topic of domain adaption (Saito et al., 2019) plays an important role, as the artificial imagery and the real-world data stem from different domains and distributions. Also, better ways of incorporating negative samples, which means the wide variety of samples that do not belong to the object, into the training data have to be developed, and their effect has to be analyzed. Semi-supervised learning methods may help to address this issue. This is closely related to out-of-distribution samples, which also need to be addressed (Hein et al., 2019). We have seen that certain aspects of the data set possibly limit and distort the real performance of the object detection network. To this end, we plan to modify the annotation of the Potsdam data set to contain the partly occluded vehicles in the annotations, and at the same time, assess the annotations of fragmented image structures. Finally, we plan to test the methods on more data sets, especially vehicle-related data sets like DLR-SkyScapes (Azimi et al., 2019).

ACKNOWLEDGEMENTS

The authors would like to thank the DVGW – German Technical and Scientific Association for Gas and Water – for funding this work as part of the research project G201819 - Antonia.

(a) Detections of the baseline model trained on the complete real-world data set. The detections are quite accurate and have high confidences; no false positives are visible.

(b) Detections of the model trained with 1000 generated images only. A lot of false positive detections are visible; however, the true positive detections are quite accurate.

(c) Detections of the model trained with 1000 generated images and eight real-world images. The model produces no false positives in the left image, and in the right image, much fewer false positives are visible.

(d) Detections of the model trained with 1000 generated images and 32 real-world images. The model produces no false positives in the left image, and in the right image, only one false positive is still visible.

Figure 8. Detections on two sample images proposed by models trained with different amounts of real-world and generated images. We applied a confidence threshold of 0.3 for all detections. In general, the models' proposals have high confidences for true positives and most of the false positives could be suppressed with a higher threshold, but it allows us to show the models' increasing ability to separate objects from the background.

Figure 9. Left: An example of detections of hidden vehicles: the detection of the car in the middle has no corresponding annotation, which counts as a false positive; the detection of the lower car is better than the annotation, which may also impact the AP negatively. Right: An example of a detection of a fragmented vehicle.

REFERENCES

Azimi, S. M., Henry, C., Sommer, L., Schumann, A., Vig, E., 2019. Skyscapes fine-grained semantic understanding of aerial scenes. *The IEEE International Conference on Computer Vision (ICCV)*.

Berthelot, D., Carlini, N., Goodfellow, I., Papernot, N., Oliver, A., Raffel, C. A., 2019. Mixmatch: A holistic approach to semi-supervised learning. *Advances in Neural Information Processing Systems*, 5050–5060.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., Fei-Fei, L., 2009. ImageNet: A Large-Scale Hierarchical Image Database. *CVPR09*.

DeVries, T., Taylor, G. W., 2017. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*.

He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Hein, M., Andriushchenko, M., Bitterwolf, J., 2019. Why relu networks yield high-confidence predictions far away from the training data and how to mitigate the problem. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 41–50.

Heipke, C., Rottensteiner, F., 2020. Deep learning for geometric and semantic tasks in photogrammetry and remote sensing. *Geo-spatial Information Science*, 1–10.

Howard, J. et al., 2018. fastai. https://github.com/fastai/fastai.

Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S., 2017a. Feature pyramid networks for object detection. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2117–2125.

Lin, T.-Y., Goyal, P., Girshick, R., He, K., Dollár, P., 2017b. Focal loss for dense object detection. *Proceedings of the IEEE international conference on computer vision*, 2980–2988.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C. L., 2014a. Microsoft coco: Common objects in context. *European Conference on Computer Vision*, 740–755.

Lin, T.-Y., Maire, M., Belongie, S. J., Bourdev, L. D., Girshick, R. B., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C. L., 2014b. Microsoft COCO: Common Objects in Context. *CoRR*, abs/1405.0312. http://dblp.uni-trier.de/db/journals/corr/corr1405.html#LinMBHPRDZ14.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., Berg, A. C., 2016. Ssd: Single shot multibox detector. *European conference on computer vision*, Springer, 21–37.

Ma, L., Liu, Y., Zhang, X., Ye, Y., Yin, G., Johnson, B. A., 2019. Deep learning in remote sensing applications: A meta-analysis and review. *ISPRS journal of photogrammetry and remote sensing*, 152, 166–177.

Movshovitz-Attias, Y., Kanade, T., Sheikh, Y., 2016. How useful is photo-realistic rendering for visual learning? *European Conference on Computer Vision*, Springer, 202–217.

Mundhenk, T. N., Konjevod, G., Sakla, W. A., Boakye, K., 2016. A large contextual dataset for classification, detection and counting of cars with deep learning. *European Conference on Computer Vision*, Springer, 785–800.

Nair, V., Alonso, J. F., Beltramelli, T., 2019. RealMix: Towards Realistic Semi-Supervised Deep Learning Algorithms. *arXiv preprint arXiv:1912.08766*.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S., 2019. Pytorch: An imperative style, high-performance deep learning library. H. Wallach, H. Larochelle, A. Beygelzimer, F. dÁlché-Buc, E. Fox, R. Garnett (eds), *Advances in Neural Information Processing Systems 32*, Curran Associates, Inc., 8024–8035.

Peng, X., Sun, B., Ali, K., Saenko, K., 2015. Learning deep object detectors from 3d models. *Proceedings of the IEEE International Conference on Computer Vision*, 1278–1286.

Richter, S. R., Vineet, V., Roth, S., Koltun, V., 2016. Playing for data: Ground truth from computer games. *European conference on computer vision*, Springer, 102–118.

Rottensteiner, F., Sohn, G., Gerke, M., Wegner, J. D., 2013. ISPRS Test Project on Urban Classification and 3D Building Reconstruction. , ISPRS - Commission III - Photogrammetric Computer Vision and Image Analysis, Working Group III / 4 - 3D Scene Analysis.

Saito, K., Ushiku, Y., Harada, T., Saenko, K., 2019. Strong-weak distribution alignment for adaptive object detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 6956–6965.

Smith, L. N., 2018. A disciplined approach to neural network hyper-parameters: Part 1 – learning rate, batch size, momentum, and weight decay.

Sun, P., Chen, G., Luke, G., Shang, Y., 2018. Salience Biased Loss for Object Detection in Aerial Images. *arXiv preprint arXiv:1810.08103*.

Tremblay, J., Prakash, A., Acuna, D., Brophy, M., Jampani, V., Anil, C., To, T., Cameracci, E., Boochoon, S., Birchfield, S., 2018. Training deep networks with synthetic data: Bridging the reality gap by domain randomization. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 969–977.

Xia, G.-S., Bai, X., Ding, J., Zhu, Z., Belongie, S., Luo, J., Datcu, M., Pelillo, M., Zhang, L., 2018. Dota: A large-scale dataset for object detection in aerial images. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3974–3983.

Yun, S., Han, D., Oh, S. J., Chun, S., Choe, J., Yoo, Y., 2019. Cutmix: Regularization strategy to train strong classifiers with localizable features. *Proceedings of the IEEE International Conference on Computer Vision*, 6023–6032.

Zhang, H., Cisse, M., Dauphin, Y. N., Lopez-Paz, D., 2017. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*.

Zhang, Z., He, T., Zhang, H., Zhang, Z., Xie, J., Li, M., 2019. Bag of freebies for training object detection neural networks. *arXiv preprint arXiv:1902.04103*.

Zheng, K., Wei, M., Sun, G., Anas, B., Li, Y., 2019. Using vehicle synthesis generative adversarial networks to improve vehicle detection in remote sensing images. *ISPRS International Journal of Geo-Information*, 8(9), 390.