Boosting Instance Segmentation with Synthetic Data: A study to overcome the limits of real world data sets

Florentin Poucin
École des Ponts Paristech
Champs-sur-Marne, France
florentin.poucin@eleves.enpc.fr

Andrea Kraus, Martin Simon
Valeo Schalter und Sensoren GmbH,
Kronach, Germany
{andrea.kraus, martin.simon}@valeo.com

Abstract

A major issue related to computer vision for the automotive industry is that real-world perception models require huge amount of well-annotated data to achieve decent performance. While this data is very expensive to collect and annotate, synthetically generated images seem to be an efficient alternative to solve this problem. More and more public data sets, composed of synthetic data, are available in various domains, however, there is too little concrete methodology to use them properly. In this paper, we propose a simple approach combining the use of synthetic and real images to boost instance segmentation. We mention some pre-processing requirements as harmonizing instance labeling and removing non-valuable instances from synthetic images. We present our training strategy based on data set mixing, and show that it overcomes the domain shift between real and synthetic data sets. A comparison study with other training approaches, such as fine-tuning techniques, highlights the benefits of our method, which boosts network performances on both real and synthetic image inferences.

1. Introduction

Deep Neural Networks (DNNs) are nowadays achieving the best performance in many tasks related to perception for autonomous driving, such as object detection or semantic segmentation. While network architecture and GPU computational power are constantly improving, the need for properly curated data has become one of the biggest issues in this field. DNNs need large amounts of training data to ensure good accuracy and robustness. Training data sets must reflect reality by properly representing frequently encountered situations, but also rare events and corner cases. Collecting and annotating real-world images that make up these data sets is very expensive. In view of these high costs, there is interest in the possibility of synthetically generating realistic data. Recent examples, such as the NVIDIA Omniverse platform [20], develop synthetic data generation algorithms for deep learning training applications. Image generation techniques drastically reduce the annotation effort to collect ground truth data, but the effectiveness of the produced images, in properly feeding DNNs, is not immediate. Many public synthetic data sets are nowadays available and represent a high potential source of data for many tasks. Our motivation is to explore the right use of synthetic data in order to boost real world automotive AI perception.

In order not to limit this work to common practices on how to properly use synthetic data, we focus on a precise use case, and highlight for it the most effective methodology to improve DNN performances throughout our experiments. We restrict the scope of our study to instance segmentation, which is about detecting and delineating each distinct object of interest appearing in an image. To perform this task, we use a Mask R-CNN [10] architecture to which we add the PointRend module [13]. As Mask R-CNN architectures are currently state-of-the-art for Instance Semantic Segmentation and several other image based perception tasks, the overall results of our experiments are applicable in a general fashion. Regarding the data, we use real-world images from the Cityscapes data set [3] and synthetically generated images from the Synscapes data set [29].

In this work, we raise different issues from the use of synthetic data to improve the performance of the chosen neural network. We propose a data set mixing approach taking advantage of both real and synthetic images. The full methodology is shown in Figure 1. Our contribution could be summarize as follows:

- We highlight pre-processing requirements related to the use of synthetic data. Ensuring high quality of rendering and annotations for synthetic images appear to be essential for the network to perform well.
- We present our data set mixing strategy, which over-

Figure 1. **Processing steps for the use of synthetic data.** On the left, the pre-processing steps: labeling harmonization regarding class definition and non-valuable synthetic instance removal. On the right, different training approaches (real-world training, mixed data set training, fine-tuning, synthetic training) and their performance obtained on Cityscapes (blue) and Synscapes (orange).

comes the current PointRend performances on both Synscapes and Cityscapes data sets.

- We give a comprehensive comparison of our method with a more common transfer learning approach based on fine-tuning.

2. Related Work

Mask R-CNN [10] models have been top ranked for many tasks, including instance segmentation, in several benchmarks e.g. [3, 16]. These region-based approaches are very effective for small instances but could show over-smoothing issues for larger objects. In order to improve this aspect, more holistic approaches using the image structure are given in [1, 2, 27]. The PointRend (*Pointbased Rendering*) module is presented in [13] as an effective alternative to Mask R-CNN's default mask head. The authors were inspired by classical ideas from computer graphics to perform point-based segmentation predictions. More recently, Liang *et al.* [15] propose PolyTransform, a model using modern polygon-based methods to improve the precision of instance masks. Adapted from language to vision, Transformer architectures [4, 17] also show promising results for instance segmentation.

A lot of research has been made about the generation of synthetic data to improve DNNs performance for real world tasks. In [22, 28], synthetic data are generated and used to overcome a lack of well labeled images in their specific field. Dwibedi *et al.* [5] generate synthetic images by cutting and pasting object instances onto diverse environment, whereas Su *et al.* [26] propose to use 3D models for the generation. Human annotated real-world data are substituted with virtual world images from the video game GTA V in [12]. In the same vein as Synscapes [29] and Cityscapes [3], Virtual KITTI [6] synthetically replicates a part of the KITTI data set [8]. Most of these papers highlight the benefits of using synthetic images to improve neural network performance. Valuable similarities between synthetic and real images are described in [21].

Seib *et al.* [25] offer a review of several approaches using synthetic data to train neural networks. Several papers present complex models, based on specific network architectures, which overcome the domain-shift between real and synthetic images. An autoencoder-approach for lane detection is mentioned in [7] and generative adversarial networks are used in [24] to achieve domain adaptation. [23] proposes to deal with foreground and background synthetic classes with two different architectures, the combination of which makes it possible to dispense with the use of real data. These complex models show good performance but require a large architectural effort, which makes them not very adaptable and efficient. Simpler methods, such as transfer learning techniques in which knowledge learned from synthetic images is used for real world tasks, also achieve a high level of performance and are widely used. ImageNet pre-training paradigm in computer vision is discussed in [9], whereas authors of [14, 19] propose to rethink hyperparameters for fine-tuning methods. Features transferability and frozen layers are detailed in [30, 11]. Some mixed training and fine-tuning issues are approached by [18].

3. Data set Alignment

3.1. Settings and metrics

The aim of this study is to achieve, with the PointRend network, the best possible performance on real-world inference. We use the baseline Cityscapes partition, 2975 images for training and 500 for validation, and the Synscapes data set includes 20k images for training and 5k for validation and test. Regarding the evaluation, we use Cityscapes and Synscapes basic metrics for instance segmentation. Instance-level performance is evaluated by an average precision on the region level for each class and averaged across a range of overlap thresholds. There are 10 different thresholds, ranging from 0.5 to 0.95 in steps of 0.05. The overlap is computed at the region level, making it equivalent to the IoU (Intersection-over-Union) of a single instance. Each evaluation, either on Cityscapes or Synscapes images, provides an average precision score for each of the 8 instance classes. These classes are "person", "rider", "car", "truck", "bus", "train", "motorcycle" and "bicycle". In this work, we mainly evaluate the performance of the network through its overall average precision, obtained by averaging the precision over the 8 classes. Except for the fine-tuning approach, we use ImageNet pretrained weights instead of random initialization in order to speed the convergence [9]. Every training is performed on 4 GPUs with the same batch size of 8 images, i.e. 2 images per GPU.

3.2. Data set Harmonization

Before involving images from Synscapes and Cityscapes in the training process, we need to ensure that the characteristics, e.g. definition of classes and instances of the labels of both data sets are same. Even if Synscapes was designed to be similar to Cityscapes, there remain some important differences in the data. For example, regarding instance definition, a rider and his vehicle (bicycle or motorcycle) are considered as two different instances in Cityscapes but as a single one in Synscapes. An example is given in figure 2, where the top image shows the labeling convention of Syncapes and the bottom image visualizes the separated instances of rider and motorcycle like it is in Cityscapes data set. This mismatch directly affects the network training which, then, tries to learn two incompatible interpretations of the couple "rider" + "motorcycle". To overcome this mismatch, we adjust the whole Synscapes data set to the Cityscapes convention in a pre-processing step.

Since the given semantic segmentation labels in Synscapes include rider, bicycle and motorcycle, an additional instance has been labeled as rider, if any of the pixels in the initial bicycle or motorcycle instance are identified as rider in the corresponding semantic segmentation. The area of the initial motorcycle or bicycle instance is then reduced to the strict vehicle area. Figure 2 shows an example of the applied instance creation in Synscapes data set. The top image visualizes the original annotated instances. A separation of rider and vehicle is achieved by using the semantic segmentation in the middle. The result, compatible with the definitions of labeling in Cityscapes data set, is shown in the lowest image.

Figure 2. **Instance label harmonization.** On top, original Synscapes labeling, rider and his vehicle compose a single instance. In the middle picture, semantic segmentation labeling contains the pixel-wise class information. At the bottom, the new labeling becomes consistent with the Cityscapes data set.

Instance definition of a rider and his vehicle is one issue related to the harmonization, but there are many others. We can mention the difference in labeling for wheelchairs or for motorcycles and bicycles without riders. As far as possible, we need to make the labeling of the two data sets consistent, in order to get the maximum value from both real and synthetic images.

Training set \ Evaluation set	Synscapes	Cityscapes
Synscapes	**31.6%**	12.9%
Cityscapes	11.8%	**35.7%**

Table 1. **Average precision obtained by the PointRend module**. Evaluation made for instance segmentation on Cityscapes and Synscapes data sets.

3.3. Overview of PointRend performances

Once annotations of the two data sets are harmonized, we process first trainings and evaluations. We train and evaluate the PointRend network on both data sets individually. The Synscapes data set is reduced to 2975 images as Cityscapes to ensure comparable amount of training data. We use baseline configuration i.e. 24k iterations with a learning rate initialized at 0.01 and decreasing by steps of 0.1× after 18k and 22k iterations. Table 1 contains the first evaluation results and gives a comparison basis for the following experiments.

When the network is trained on Synscapes, it shows good performance, **31.6%**, on this precise data set but very poor performance on Cityscapes, **12.9%**. Similarly, when the network is trained on Cityscapes, its performance on Cityscapes, **35.7%**, is 3 times better than on Synscapes, **11.8%**. These first results highlight the domain shift between the two data sets. Synthetic data are not sufficient to achieve a decent performance on real world inference. A training strategy combining synthetic and real images is necessary to achieve better results on both data sets.

4. Methodology

4.1. Removing non-valuable synthetic instances

Synscapes images are very realistic and show high quality instance-level annotations. These annotations are automatically generated during the rendering of the synthetic images, every instance is accurately labeled even when it is truncated, occluded or very deep in the image background. Comprehensive labeling is made possible by the synthetic nature of these images. Although exact labeling is desirable in theory, a comparative study of the synthetic data set with the target data set (Cityscapes) showed that the number of instances including only a few pixels differ a lot.

Having the exact position information of all instances is a good point, however there is almost too much information in synthetic images. Very small instances composed of few pixels due to their occlusion or depth are present. They are almost invisible to the naked eye but still annotated, their value for DNNs training is questionable. In order to evaluate the value of the Synscapes instance labeling, we want to measure the impact of having more or less labeled instances in the images. If instances are considered valuable, removing their label in the training images should decrease the performance of the network on the validation data set. The choice of the labels to be removed is based on a size criterion, i.e. the number of pixels representing the labeled instance in the image. This size criterion indirectly contains truncation, occlusion and depth notions. Removal thresholds percentage are introduced for each class. For example, applying a 30% removal threshold for cars means removing 30% of car labels in the data set, starting with the smallest instances in terms of pixel representation. The same image, from the Synscapes data set, is depicted with six different removal thresholds in Figure 3.

We create 10 reduced versions of the Synscapes data set by applying to the same 2975 synthetic images different instance removal thresholds from 0% to 90%. In order to evaluate the impact of label removal on learning effectiveness, the network is trained on the 10 different data sets. Figure 4 represents, for each class, the average precision obtained on Synscapes against the removal threshold applied to the training set. Precision trend is quite the same for every class and can be separated into two phases. First, for small removal thresholds, the precision is constant or even slightly increasing. This means that for each class, removing the smallest instances does not have a negative effect on the final performance and may even improve it. Then, there is a strong decrease of the average precision when the removal threshold becomes too high. This can be explained by a poor feeding of the DNN due to a lack of training labels. The limit threshold between the two phases depends on the class. For most of them, the limit threshold is around 30% but regarding "person" class it is rather around 60%.

This experiment shows that the label value for the DNN training is not the same for all instances. For each class, removing labels from the smallest instances has beneficial effects, or at least no effect, on the network performance for Synscapes evaluation. We can consider that around 40% of the labels present on synthetic images are not valuable, and we choose to remove them from the Synscapes data set. Concerning the 2975 images of this experiment, we go from 137,255 labeled instances to 70,744. In comparison, the Cityscapes data set contains 54,060 labeled instances for the same number of images. This removal initially has 3 beneficial aspects: i) it slightly improves the network precision on synthetic images, ii) it reduces considerably the number of labeled instances and thus reduces the computational effort, and iii) it makes Synscapes and Cityscapes labelings more similar. Table 2 illustrates the benefits of this pre-processing step for the different approaches that we present in the following.

4.2. Data set mixing approach

With the pre-processing done, we present in this section our data set mixing strategy and show its performance. The

Figure 3. **Synscapes labeling for different removal thresholds.** We represent the same synthetic image and its labeling, applying 6 different removal thresholds to each class individually: 0%, 40%, 60% on top, and 70%, 80%, 90% at the bottom.

Figure 4. **Synscapes evaluation score when removing training instances.** Class precision scores obtained by the network against the instance removal threshold applied to the training set

Training approach	Evaluation	Instance removal: off	Instance removal: on
Pre-training on Synscapes	Synscapes	38.6	**42.0**
	Cityscapes	**14.9**	14.2
Mixing Synscapes and Cityscapes	Synscapes	38.6	**41.0**
	Cityscapes	38.9	**40.9**
Fine-tuning on Cityscapes	Synscapes	28.7	**29.6**
	Cityscapes	37.6	**39.6**

Table 2. **Impact of instance removal pre-processing on final precision scores.** For almost all training approaches, the final average precision obtained on both data sets, is better with instance removal pre-processing than without.

principle of our approach is simple, but effective. Since we managed to make our two data sets as similar as possible with the methods presented, there is no more concrete incompatibility, for the network, to learn both synthetic and real images at the same time. Although domain shift is still a reality, each image, real or synthetic, contains valuable features for the network performance and robustness. For these reasons, we mix real and synthetic images together, to create a single training data set.

Figure 5. **Impact of synthetic/real image ratio on network performance.** The average precision obtained on Cityscapes, in blue, and on Synscapes, in orange, is plotted as a function of the synthetic/real ratio of the training images

During the training, we do not make any process distinction between real and synthetic images. However, we do make this distinction during the creation of the mixed data set. The latter requires finding the right composition, i.e. the most relevant distribution between real and synthetic images. The Synscapes data set is much larger than the Cityscapes data set, but it is not necessarily relevant to use both data sets in their entirety. Therefore, we create 11 different data sets of 2975 images, each with a different percentage of real images, from 0% to 100%. The images has been randomly sampled from the given training data sets. Then, we evaluate the impact of data set composition, i.e. synthetic/real images ratio, on network performance.

Figure 5 illustrates the performance of the network when

Training epochs	Amount of additional synthetic data					
	0k	3k	6k	9k	12k	20k
33	32.6	37.6	38.4	38.8	38.9	40.3
50	36.2	37.9	38.9	39.3	39.3	**40.9**
66	35.7	37.2	37.7	39.3	38.9	40.5

Table 3. **Average precision scores obtained on Cityscapes with a data set mixing approach**. Precision score is given against the number of training epochs and the amount of synthetic images used in addition to the 2975 Cityscapes images.

it is trained on the customized data sets and evaluated on Cityscapes and Synscapes. The blue curve and the orange curve, respectively representing Cityscapes evaluation and Synscapes evaluation, have basically reversed trends. The more training images from a specific data set, the better the performance in the evaluation on that data set. It is a trivial observation but shows an essential similarity between synthetic and real feature learning. Synthetic data has not the same value as real data, when the objective is to perform on real data, and vice versa. We can also observe a large performance gap in Cityscapes evaluation between 0% and 10%, and another performance gap in Synscapes evaluation between 90% and 100%. The performance in Cityscapes evaluation, shown in blue, almost doubles between 0% and 10%. This improvement in accuracy, due to a 10% synthetic to real image replacement, is greater than the improvement due to replacing the remaining 90%. This experiment mainly shows two results. First, real and synthetic data differ in nature, the gaps of performance highlight the necessity to use both of them for the training, even in unequal proportions. Furthermore, the monotonicity of the curves prove that, if the objective is to perform on real images, it is always preferable to feed the network with a real image rather than a synthetic image. The opposite result is also true when the objective is to perform on synthetic images. Regarding our study, the most important seems to use the entire Cityscapes data set, while the amount of additional Synscapes images remains to be determined.

Table 3 contains the performance of the network obtained when evaluating on Cityscapes, against the amount of additional Synscapes images involved and the number of epochs set for the training. For a fixed data set composition, we observe that it is generally preferable to train the network on 50 epochs. With respect to our experimental setup, increasing or decreasing this number leads to a decrease in performance. Otherwise, for a fixed number of epochs, the more images the data set contains, the better the performance. These results show that the real/synthetic image ratio has very little impact on the performance as long as enough epochs are applied to the training. The network obtains a performance up to **40.9%** for a 50 epoch training on a mixed data set containing Cityscapes and Synscapes

Iteration limitation	Best average precision (%)	Composition (real/synthetic)
20k	35.1	3k/0k
40k	38.4	3k/6k
60k	38.9	3k/6k
80k	39.3	3k/9k
100k	40.3	3k/20k
150k	40.9	3k/20k

Table 4. **Efficient composition for a limited number of iterations.** For several iteration limitations (left column), we report the best performance achieved on Cityscapes evaluation (middle column), and its corresponding data set composition (right column).

in their entirety. Our strategy overcomes the current performance of the PointRend module on Cityscapes validation set, average precision increased to **40.9%** from **36.2%**. Same improvement is observed on Cityscapes test set, for which the network obtains a **30.5%** average precision with baseline configuration compare to **34.8%** with our data set mixing strategy. Otherwise, table 2 highlights the benefits of non-valuable instance removal for this data set mixing method. This pre-processing boosts the network performances on both Synscapes and Cityscapes evaluations.

Training a network on 23k images during 50 epochs with a batch size of 8 images makes more than 140k iterations. This computational cost raises the question of the efficiency of the method. Table 4 contains best precision scores obtained on real data when limiting the number of iterations for the training. Under 20k iterations, which approximately correspond to 50 epochs on 3k images for our batch size, adding synthetic data to the data set seems ineffective. When the number of allowed iterations increases, it becomes interesting to add synthetic images. In general, we find that the best trade-off, between performance and training time, is obtained for a data set size enabling to spend 50 epochs on diverse images. An effective strategy would be to adapt the amount of additional synthetic images as a function of the number of iteration allowed. For example, if for any reason we can spend a maximum of 80k iterations for our training, it corresponds to 640k processed images, to 50 epochs on 12,800 images, then it would be effective to train the network on a mixed data set composed of 3k real images and 9800 synthetic images.

4.3. Comparison study

Fine-tuning is a transfer learning technique usually employed to solve domain adaptation issues. It consists of using final weights from a pre-training model as initialization for a second training. Knowledge, learned from the source domain during pre-training, generally improves the learning of features from the target domain. In this study, Synscapes and Cityscapes are respectively the source and the

target domains. In this section, we use fine-tuning to improve the performance of our network and compare the results obtained with the performance achieved with a data set mixing strategy. We present our different steps to optimize the fine-tuning approach and summarize the main results in table 5.

First step is to train the network on synthetic data and get the valuable weights to initialize the second training. We run 100 epochs on the 20k Synscapes images, which have been harmonized and pre-processed by removing non-valuable instances. After this pre-training, the network achieves an average precision of **42.0%** when evaluating on Synscapes. There is a large improvement compared to the initial training on 2975 Synscapes images run in subsection 3.3. This result highlights the benefits of using Synscapes in its entirety, increasing the number of epochs, and pre-processing the images. The performance on real images, **14.2%**, remains very poor.

Second step is to fine-tune on real images. We use baseline configuration as 3.3 except that we initialize the model with Synscapes pre-trained weights rather than ImageNet pre-trained weights. After this second training, the network precision decreases to **19.7%** when evaluating on Synscapes, but increases to **38.0%** when evaluating on Cityscapes. Compared to baseline training, replacing the initialization weights and fine-tuning already improve the average precision on real images to **38.0%** from **35.7%**.

In this work, fine-tuning is about learning knowledge from real images, adapting the one learned from synthetic images. We want to learn additional content without losing the generalization ability provided by pre-trained weights. During the second training, controlling hyperparameters, like learning rate and momentum, which directly supervise the way and the speed of network learning, is crucial. When the learning rate is too high, weight variation between each iteration is too important and initialization with specific weights becomes useless. In the opposite, with a too small learning rate, weights are not sufficiently adjusted to the training data set, additional content is simply not learned. After a pre-training on Synscapes, we fine-tune the network on Cityscapes with different combinations of learning rate and momentum.

Figure 6 illustrates the average precision obtained by the network against hyperparameter settings. The x-axis is logarithmic and corresponds to the value of the learning rate η. This value is the initialization value of the learning rate, which is reduced by 0.1x, 2 times during the training, in the last 60k and last 20k iterations. When η is above 0.02 the model diverges and when it is under 0.00005 the model converges too slowly, making the precision very low. We only use two different values for the momentum, namely 0.0 and 0.9. The two curves, orange for a zero momentum and blue for a momentum equal to 0.9, respectively reach their maximum for a learning rate equal to 0.0005 and 0.005. The basic value of learning rate and momentum are $\eta = 0.01$ and $m = 0.9$. In this case, a lower learning rate enables a better fine-tuning performance. Otherwise, overlapped by factor of 10 along the x-axis, the two curves have approximately the same trend. This can be related to a result from [14], showing that the fine tuning performance depends on an effective learning rate defined by $\eta' = \eta/(1-m)$ with η the learning rate and m the momentum. The difference of momentum between 0.0 and 0.9 corresponds to a factor of 10 between the effective learning rates. The maximum of the two curves are reached for the same effective learning rate of 0.005, which is lower than its baselines value of 0.1. There are two options to reach the optimal precision : reduce the learning rate significantly or reduce the learning rate and the momentum simultaneously. For $\eta = 0.005$ and $m = 0.0$, we obtain an average precision up to **29.6%** on Synscapes and up to **39.6%** on Cityscapes. Consistent learning rate and momentum enable to control the fine-tuning phase improving the performance on both data sets at the same time.

In this work, fine-tuning strategy achieves an average precision up to **39.6%** when evaluating on Cityscapes, which overcomes the baseline PointRend performance. We mention good usage for pre-processing and pre-training on

Applied methods	Evaluation set: Synscapes	Evaluation set: Cityscapes
Basic train on Cityscapes	11.8	35.7
Pre-train on 3k Synscapes images	31.6	12.9
+ Retrain with 20k Synscapes images	38.6	14.9
+ Remove non-valuable instances	**42.0**	14.2
+ Fine-tune on Cityscapes	19.7	38.0
+ Optimize hyperparameters	29.6	**39.6**

Table 5. **Precision scores related to fine-tuning.** This table contains the network performance obtained for the different steps of the fine-tuning approach and highlights the benefits of each of them ("+" means in addition to previous steps).

Figure 6. **Learning rate and Momentum impact on training.** Average precision score from Cityscapes evaluation is plotted as a function of training learning rate (logarithmic x-axis) and momentum (curve color).

Synscapes and we highlight optimal learning rate and momentum settings for fine-tuning. Once again, table 2 illustrates the benefits of the synthetic instance removal for this method. Fine-tuning technique achieves very good results while keeping the two data sets separated and independent. Once computed, Synscapes pre-trained weights can be used for several configuration involving different target domains or different networks. This specific generalization ability of fine-tuning is very interesting and makes this approach very popular. However, we have shown in this study that the performance obtained by a data set mixing strategy remains better on both data sets than by using fine-tuning.

5. Conclusion

To conclude, synthetic data shows promising properties to boost the performance of DNNs on real-world instance segmentation. Although synthetic data is not yet able to fully replace real data for training, it can easily overcome a lack of real-world images and can be used to improve neural network performance. In this work, we have presented some pre-processing requirements and a training strategy comparison related to the use of synthetic data. Figure 1 recaps the main points discussed in this paper. We have highlighted the benefits of harmonizing and pre-processing synthetic images before using them. Removing the non-valuable instances from synthetic data reduces the computational effort and improves the precision on both synthetic and real data sets. Once the data have been made as similar as possible, we propose a training strategy mixing them into a single data set, for which real and synthetic images are processed in parallel the same way. We have evaluated the impact of the mixed data set composition, and we conclude that real images remain the most valuable data but additional synthetic images can significantly boost the network performance. Our approach overcomes the current performance of the PointRend module trained on real images only. We have also compared our training strategy to a common transfer learning approach consisting in pre-training on synthetic data and fine-tuning on real data. This second method achieves a very good performance, especially when controlling the fine-tuning phase with consistent hyperparameters, and shows interesting generalization ability. However, the performance of the data set mixing strategy remains better on both Cityscapes and Synscapes data sets. Some parts of these results naturally depend on the choice we made regarding the network, the task performed and the two data sets. However, many elements of this study, such as pre-processing and data set mixing principles, could be very useful for other application scenarios involving synthetic data.

References

[1] Anurag Arnab and Philip HS Torr. Pixelwise instance segmentation with a dynamically instantiated network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 441–450, 2017. 2

[2] Xinlei Chen, Ross Girshick, Kaiming He, and Piotr Dollár. Tensormask: A foundation for dense object segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2061–2069, 2019. 2

[3] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016. 1, 2

[4] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *arXif preprint arXiv:2010.11929*, 2020. 2

[5] Debidatta Dwibedi, Ishan Misra, and Martial Hebert. Cut, paste and learn: Surprisingly easy synthesis for instance detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1301–1310, 2017. 2

[6] Adrien Gaidon, Qiao Wang, Yohann Cabon, and Eleonora Vig. Virtual worlds as proxy for multi-object tracking analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4340–4349, 2016. 2

[7] Noa Garnett, Roy Uziel, Netalee Efrat, and Dan Levi. Synthetic-to-real domain adaptation for lane detection. In *Proceedings of the Asian Conference on Computer Vision*, 2020. 2

[8] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013. 2

[9] Kaiming He, Ross Girshick, and Piotr Dollár. Rethinking imagenet pre-training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4918–4927, 2019. 2, 3

[10] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 1, 2

[11] Stefan Hinterstoisser, Vincent Lepetit, Paul Wohlhart, and Kurt Konolige. On pre-trained image features and synthetic images for deep learning. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pages 0–0, 2018. 2

[12] Matthew Johnson-Roberson, Charles Barto, Rounak Mehta, Sharath Nittur Sridhar, Karl Rosaen, and Ram Vasudevan. Driving in the matrix: Can virtual worlds replace human-generated annotations for real world tasks? In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 746–753. IEEE, 2017. 2

[13] Alexander Kirillov, Yuxin Wu, Kaiming He, and Ross Girshick. Pointrend: Image segmentation as rendering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9799–9808, 2020. 1, 2

[14] Hao Li, Pratik Chaudhari, Hao Yang, Michael Lam, Avinash Ravichandran, Rahul Bhotika, and Stefano Soatto. Rethinking the hyperparameters for fine-tuning. In *International Conference on Learning Representations*, 2020. 2, 7

[15] Justin Liang, Namdar Homayounfar, Wei-Chiu Ma, Yuwen Xiong, Rui Hu, and Raquel Urtasun. Polytransform: Deep polygon transformer for instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9131–9140, 2020. 2

[16] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 2

[17] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *arXiv preprint arXiv:2103.14030*, 2021. 2

[18] Farzan Erlik Nowruzi, Prince Kapoor, Dhanvin Kolhatkar, Fahed Al Hassanat, Robert Laganiere, and Julien Rebut. How much real data do we actually need: Analyzing object detection performance using synthetic and real data. *arXiv preprint arXiv:1907.07061*, 2019. 2

[19] Biserka Petrovska, Tatjana Atanasova-Pacemska, Roberto Corizzo, Paolo Mignone, Petre Lameski, and Eftim Zdravevski. Aerial scene classification through fine-tuning with adaptive learning rates and label smoothing. *Applied Sciences*, 10(17):5792, 2020. 2

[20] NVIDIA Omniverse Platform. https://developer.nvidia.com/nvidia-omniverse-platform. Accessed: 2021-08-09. 1

[21] Artem Rozantsev, Vincent Lepetit, and Pascal Fua. On rendering synthetic images for training an object detector. *Computer Vision and Image Understanding*, 137:24–37, 2015. 2

[22] Christos Sakaridis, Dengxin Dai, and Luc Van Gool. Semantic foggy scene understanding with synthetic data. *International Journal of Computer Vision*, 126(9):973–992, 2018. 2

[23] Fatemeh Sadat Saleh, Mohammad Sadegh Aliakbarian, Mathieu Salzmann, Lars Petersson, and Jose M Alvarez. Effective use of synthetic data for urban scene semantic segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 84–100, 2018. 2

[24] Swami Sankaranarayanan, Yogesh Balaji, Arpit Jain, Ser Nam Lim, and Rama Chellappa. Learning from synthetic data: Addressing domain shift for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3752–3761, 2018. 2

[25] Viktor Seib, Benjamin Lange, and Stefan Wirtz. Mixing real and synthetic data to enhance neural network training–a review of current approaches. *arXiv preprint arXiv:2007.08781*, 2020. 2

[26] Hao Su, Charles R Qi, Yangyan Li, and Leonidas J Guibas. Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2686–2694, 2015. 2

[27] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2):154–171, 2013. 2

[28] Daniel Ward, Peyman Moghadam, and Nicolas Hudson. Deep leaf segmentation using synthetic data. In *British Machine Vision Conference*, 2018. 2

[29] Magnus Wrenninge and Jonas Unger. Synscapes: A photorealistic synthetic dataset for street scene parsing. *arXiv preprint arXiv:1810.08705*, 2018. 1, 2

[30] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Proceedings of the 27th International Conference on Neural Information Processing Systems-Volume 2*, pages 3320–3328, 2014. 2