Chapter 9
Directions in Synthetic Data Development

In this chapter, we outline the main directions that we believe to represent promising ways to further improve synthetic data, making it more useful for a wide variety of applications in computer vision and other fields. In particular, we discuss the idea of domain randomization (Section 9.1) intended to improve the applications of synthetic datasets, methods to improve CGI-based synthetic data generation itself (Section 9.2), ways to create synthetic data from real images by cutting and pasting (Section 9.3), and finally possibilities to produce synthetic data by generative models (Section 9.4). The latter means generating useful synthetic data from scratch rather than domain adaptation and refinement, which we consider in a separate Chapter 10.

9.1 Domain Randomization

Domain randomization is one of the most promising approaches to make straightforward transfer learning from synthetic-to-real data actually work. The basic idea of domain randomization had been known since the 1990s [394], but was probably first explicitly presented and named in [861]. Consider a model that is supposed to train on a synthetic dataset $D_{\text{syn}} \sim p_{\text{syn}}$, where p_{syn} denotes the distribution of synthetic data, and later be applied to a real dataset $D_{\text{real}} \sim p_{\text{real}}$, where p_{real} is the distribution of real data. The idea is simple: let us try to make the synthetic data distribution p_{syn} sufficiently wide and varied so that the model trained on p_{syn} will be robust enough to work well on p_{real}.

Ideally, we would like to cover p_{real} with p_{syn}, but in reality this is never achieved directly. Instead, synthetic data in computer vision can be randomized and made more diverse in a number of different ways at the level of either constructing a 3D scene or rendering 2D images from it

S. I. Nikolenko, *Synthetic Data for Deep Learning*, Springer Optimization and Its Applications 174, https://doi.org/10.1007/978-3-030-75178-4_9

Fig. 9.1 Sample images generated by the domain randomization approach by Tremblay et al. [867] for an outdoor driving dataset.

- at the scene construction level, a synthetic data generator (SDG) can randomize the number of objects, its relative and absolute positions, number and shape of distractor objects, contents of the scene background, textures of all objects participating in the scene, and so on;
- at the rendering level, SDG can randomize lighting conditions, in particular, the position, orientation, and intensity of light sources, change the rendering quality by modifying image resolution, rendering type such as ray tracing or other options, add random noise to the resulting images, and so on.

Tobin et al. [861] made the first steps to show that domain randomization works well; they used simple geometric shapes (polyhedra) as both target and distractor objects, random textures such as gradient fills or checkered patterns. The authors found that synthetic pretraining is indeed very helpful when only a small real training set is available, but helpful only if sufficiently randomized, in particular, when using a large number of random textures.

This approach was subsequently applied to a more ambitious domain by NVIDIA researchers Tremblay et al. [867], who trained object detection models on synthetic data with the following procedure:

- create randomized 3D scenes, adding objects of interest on top of random surfaces in the scenes;
- add so-called "flying distractors", diverse geometric shapes that are supposed to serve as negative examples for object detection;
- add random textures to every object, randomize the camera settings, lighting, and other parameters.

The resulting images are completely unrealistic (see Fig. 9.1 for a few samples that are supposed to represent outdoor scenes to train the detection of cars), yet diverse enough that the networks have to concentrate on the shape of the objects in question. Tremblay et al. report improved car detection results for R-FCN [178] and SSD [539] architectures (but failing to improve Faster R-CNN [719]) on their dataset compared to Virtual KITTI (see Section 7.2), as well as improved results on hybrid datasets (adding a domain-randomized training set to COCO [525]), a detailed ablation study, and extensive experiments showing the effect of various hyperparameters.

Since then, domain randomization has been used and further developed in many works. Borrego et al. [85] aim to improve object detection for common objects, showing that domain randomization in the synthetic part of the dataset significantly

improves the results. Tobin et al. [860] consider robotic grasping, a problem where the lack of real data is especially dire (see also Sections 7.4 and 10.6). They use domain randomization to generate a wide variety of unrealistic procedurally generated object meshes and textured objects for grasping, so that a model trained on them would generalize to real objects as well. They show that a grasping model trained entirely on non-realistic procedurally generated objects can be successfully transferred to realistic objects.

Up until recently, domain randomization had operated under the assumption that realism is not necessary in synthetic data. Prakash et al. [684] take the next logical step, continuing this effort to *structured* domain randomization. They still randomize all of the settings mentioned above, but only within realistic ranges, taking into account the structure and context of a specific scene.

Finally, another important direction is learning *how* to randomize. Van Vuong et al. [897] provide one of the first works in this direction, concentrating on picking the best possible domain randomization parameters for sim-to-real transfer of reinforcement learning policies. They show that the parameters that control sampling over Markov decision processes are important for the quality of transferring the learned policy to a real environment and that these parameters can be optimized. We mark this as a first attempt and expect more works devoted to structuring and honing the parameters of domain randomization.

9.2 Improving CGI-Based Generation

The basic workflow of synthetic data in computer vision is relatively straightforward: prepare the 3D models, place them in a controlled scene, set up the environment (camera type, lighting etc.), and render synthetic images to be used for training. However, some works on synthetic data present additional ways to enhance the data not by domain adaptation/refinement to real images (we will discuss these approaches in Section 10), but directly on the stage of CGI generation.

There are two different directions for this kind of added realism in CGI generation. The first direction is to make more realistic objects. For example, Wang et al. [904] recognize retail items in a smart vending machine; to simulate natural deformations in the objects, they use a surface-based mesh deformation algorithm proposed in [906], introducing and minimizing a global energy function for the object's mesh that accounts for random deformations and rigidity properties of the material (Wang et al. also use GAN-based refinement, see Section 10.3). Another approach, initiated by Rozantsev et al. [738], is to estimate the rendering parameters required to synthetize similar images from data; this approach ties into the synthetic data generation feedback loop that we discuss in Section 12.2.

The second direction is to make more realistic "sensors", introducing synthetic data postprocessing that mimics the noise characteristics of real cameras/sensors. For example, we discussed *DepthSynth* by Planche et al. [677] (see Section 6.5), a system that makes simulated depth data more realistic, more similar to real depth

sensors, while the OVVV system by Taylor et al. [853] (Section 6.6), and the ICL-NUIM dataset by Handa et al. [321] (Section 7.2) take special care to simulate the noise of real cameras. There is even a separate area of research completely devoted to better modeling of the noise and distortions in real-world cameras [72]

Apart from added realism on the level of images, there is also the question of high-level coherence and realism of the scenes. While there is no problem with coherence when the scenes are done by hand, the scale of modern datasets requires to automate scene composition as well. We note a recent joint effort in this direction by NVIDIA, University of Toronto, and MIT: Kar et al. [433] present *Meta-Sim*, a general framework that learns to generate synthetic urban environments (see also Section 7.2). *Meta-Sim* represents the composition of a 3D scene with a *scene graph* and a *probabilistic scene grammar*, a common representation in computer graphics [1026]. The goal is to learn how to transform samples coming from the probabilistic grammar so that the distribution of synthetic scenes becomes similar to the distribution of scenes in a real dataset; this is known as bridging the *distribution gap*. What's more, *Meta-Sim* can also learn these transformations with the objective of improving the performance of networks trained on the resulting synthetic data for a specific task such as object detection (see also Section 12.2).

There are also a number of domain-specific developments that improve synthetic data generation for specific fields. For example, Cheung et al. [145] present *LCrowdV*, a generation framework for crowd videos that combines a procedural simulation framework that concentrates of movements and human behaviour and a rendering framework for image/video generation, while Anderson et al. [20] develop a method for stochastic sampling-based simulation of pedestrian trajectories (see Section 6.6).

In general, while computer graphics is increasingly using machine learning to speed up rendering (by, e.g., learning approximations to complex computationally intensive transformations [424, 651]) and improve the resulting 3D graphics, works on synthetic data seldom make use of these advances; a need to improve CGI-based synthetic data is usually considered in the direction of making it more realistic with refinement models (see Section 10.1). However, we do expect further interesting developments in specific domains, especially in situations where the characteristics of specific sensors are important (such as, e.g., LIDARs in autonomous vehicles).

9.3 Compositing Real Data to Produce Synthetic Datasets

Another notable line of work that, in our opinion, lies at the boundary between synthetic data and data augmentation is to use combinations and fusions of different real images to produce a larger and more diverse set of images for training. This does not require the use of CGI for rendering the synthetic images, but does require a dataset of real images.

Early works in this direction were limited by the quality of segmentation needed to cut out real objects. For some problems, however, it was easy enough to work. For example, Eggert et al. [221] concentrate on company logo detection. To generate

synthetic images, they use a small number of real base images where the logos are clearly visible and supplied with segmentation masks, apply random warping, color transformations, and blurring, and then paste the modified (segmented) logo onto a new background image. Training on this extended dataset yielded improvements in logo detection results. In Section 6.6, we have discussed the "Frankenstein" pipeline for compositing human faces [360].

The field started in earnest with the *Cut, Paste, and Learn* approach by Dwibedi et al. [213], which is based on the assumption that only *patch-level realism* is needed to train, e.g., an object detector. They take a collection of object instance images, cut them out with a segmentation model (assuming that the instance images are simple enough that segmentation will work almost perfectly), and paste them onto randomized background scenes, with no regard to preserving scale or scene composition. Dwibedi et al. compare different classical computer vision blending approaches (e.g., Gaussian and Poisson blending [669]) to alleviate the influence of boundary artifacts after the paste; they report improved instance detection results. The work on cut-and-paste was later extended with GAN-based models (used for more realistic pasting and inpainting) and continued in the direction of unsupervised segmentation by Remez et al. [716] and Ostyakov et al. [648].

Subsequent works extend this approach for generating more realistic synthetic datasets. Dvornik et al. [212] argue that an important problem for this type of data augmentation is to preserve visual context, i.e., make the environment around the objects more or less realistic. They describe a preliminary experiment where they placed segmented objects at completely random positions in new scenes and not only did not see significant improvements for object detection on the VOC'12 dataset, but actually saw the performance deteriorate, regardless of the distractors or strategies used for blending and boundary artifact removal. Therefore, they added a separate model (also a CNN) that predicts what kind of objects can be placed in a given bounding box of an image from the rest of the image with this bounding box masked out; then the trained model is used to evaluate potential bounding boxes for data augmentation, choose the ones with the best object category score, and then paste a segmented object of this category in the bounding box. The authors report improved object detection results on VOC'12.

Wang et al. [903] develop this into an even simpler idea of *instance switching*: let us switch only instances of the same class between different images in the training set; in this way, the context is automatically right, and shape and scale can also be taken into account. Wang et al. also propose to use instance switching to adjust the distribution of instances across classes in the training set and account for class importance by adding more switching for classes with lower scores. The resulting PSIS (Progressive and Selective Instance Switching) system provides improved results on the MS COCO dataset for various object detectors including Faster-RCNN [719], FPN [523], Mask R-CNN [327], and SNIPER [803].

For a detailed consideration, let us consider a recent work by Jin and Rinard [402] who take this basic cut-and-paste approach to the next level. In essence, they still use the same basic pipeline:

- take an object space O consisting of synthetic objects placed in random poses and subjected to a number of different augmentations;
- take a context space C consisting of background images;
- superimpose objects from O against backgrounds from C at random;
- train a neural network on the resulting composite images.

However, Jin and Rinard consider this approach in detail and introduce several important tricks that allow this simple approach to provide some of the very best results available in domain adaptation and few-shot learning.

First, the sampling. One common pitfall of computer vision is that when you have relatively few examples of a class, they cannot come in a wide variety of backgrounds. Hence, in a process akin to overfitting the networks might start learning the characteristic features of the backgrounds rather than the objects in this class.

What is the easiest way out of this? How can we tell the classifier that it's the object that's important and not the background? With synthetic images, it's easy: let us place several different objects on the same background! Then, since the labels are different, the classifier will be forced to learn that backgrounds are not important and it is the objects that differentiate between classes. Therefore, Jin and Rinard take care to introduce *balanced sampling* of objects and backgrounds. The basic procedure samples a random biregular graph so that every object is placed on an equal number of backgrounds and vice versa, every background is used with the same number of objects.

The other idea used by Jin and Rinard stems from the obvious fact that the classifier must learn to distinguish between different objects. Therefore, it would be beneficial for training to concentrate on the hard cases where the classifier might confuse two objects. In [402], this idea comes in two flavors. First, specifically for images the authors suggest to superimpose one object on top of another, so that the previous object provides a maximally confusing context for the next one. Second, they use *robustness training*, a method basically equivalent to self-adversarial training that we discussed in Section 3.4 but applied to synthetic images here. The idea is that if we are training on synthetic image that might look a little unrealistic and might not be hard enough to confuse even an imperfect classifier, we can try to make it harder for the classifier by turning it into an adversarial example.

With all these ideas combined, Jin and Rinard obtain a relatively simple pipeline that is able to achieve state-of-the-art results by training with only *a single synthetic image* of each object class. Note that there is no complex domain adaptation here: all ideas can be thought of as smart augmentations similar to the ones we considered in Section 3.4.

With the development of conditional generative models, this field has blossomed into more complex conditional generation, usually called *image fusion*, that goes beyond cut-and-paste; we discuss these extensions in Section 10.4.

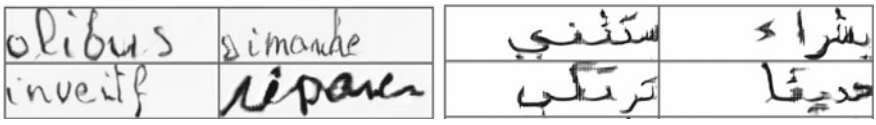
9.4 Synthetic Data Produced by Generative Models

Generative models, especially generative adversarial networks (GAN) [290] that we will discuss in detail in Chapter 4, are increasingly being used for domain adaptation, either in the form of refining synthetic images to make them more realistic or in the form of "smart augmentation", making nontrivial transformations on real data. We discuss these techniques in Chapter 10. Producing synthetic data directly from random noise for classical computer vision applications generally does not sound promising: GANs can only try to approximate what is already in the data, so why can't the model itself do it? However, in a number of applications synthetic data produced by GANs directly from random noise, usually with an abstract condition such as a segmentation mask, can help; in this section, we consider several examples of these approaches.

Counting (objects on an image) is a computer vision problem that, formally speaking, reduces to object detection or segmentation but in practice is significantly harder: to count correctly the model needs to detect all objects on the image, missing not a single one. Large datasets are helpful for counting, and synthetic data generated with a GAN conditioned on the number of objects or a segmentation mask with known number of objects, either produced at random or taken from a labeled real dataset, proves to be helpful. In particular, there is a line of work that deals with leaf counting on images of plants: ARIGAN by Giuffrida et al. [278] generates images of arabidopsis plants conditioned on the number of leaves, Zhu et al. generate the same conditioned on segmentation masks [1028], and Kuznichov et al. [490] generate synthetically augmented data that preserves the geometric structure of the leaves; all works report improved counting.

Santana and Hotz [758] present a generative model that can learn to generate realistic looking images and even videos of the road for potential training of self-driving cars. Their model is a VAE+GAN autoencoder based on the architecture from [497] that is combined with a recurrent transition model that learns realistic transitions in the embedded space. The resulting model produces synthetic videos that preserve road texture, lane markings, and car edges, keeping the road structure for at least 100 frames of the video. This interesting approach, however, has not yet led to any improvements in the training of actual driving agents.

It is hard to find impressive applications where synthetic data is generated purely from scratch by generative models; as we have discussed, this may be a principled



(a) (b)

Fig. 9.2 Sample handwritten text generated by Alonso et al. [14]: (a) French; (b) Arabic.

limitation. Still, even a small amount of additional supervision may do. For example, Alonso et al. [14] consider adversarial generation of handwritten text (see also Section 6.7). They condition the generator on the text itself (sequence of characters), generate handwritten instances for various vocabulary words, and augment the real RIMES dataset [299] with the resulting synthetic dataset (Fig. 9.2). Alonso et al. report improved character recognition performance in terms of both edit distance and word error rate. This example shows that synthetic data does not need to involve complicated 3D modeling to work and improve results; in this case, all information Alonso et al. provided for the generative model was a vocabulary of words.

A related but different field considers unsupervised approaches to segmentation and other computer vision problems based on adversarial architectures, including learning to segment via cut-and-paste [716], unsupervised segmentation by moving objects between pairs of images with inpainting [648], segmentation learned from unannotated medical images [1011], and more [70]. While this is not synthetic data *per se*, in general we expect unsupervised approaches to computer vision to be an important trend in the use of synthetic data.

At this point, we have seen many examples and applications of synthetic data. Most synthetic data generation that we have encountered has involved manual components: for instance, in computer vision, the 3D scene is usually set up by hand, with manually crafted 3D objects. However, we have already seen a few cases where synthetic data can be produced automatically with generative models. What's even more important in the context of synthetic data applications, generative models can help adapt synthetic data to make it more realistic, or adapt models for downstream tasks to work well on real data after training on synthetic. We have already introduced generative models and specifically GAN-based architectures in Chapter 4, and in the next chapter, it is time to put them to work for synthetic-to-real domain adaptation.