Synthetic Data Generation using Imitation Training

Aman Kishore
NVIDIA
amkishore@nvidia.com

Tae Eun Choe
NVIDIA
tchoe@nvidia.com

Junghyun Kwon
NVIDIA
junghyunk@nvidia.com

Minwoo Park
NVIDIA
minwoop@nvidia.com

Pengfei Hao
NVIDIA
phao@nvidia.com

Akshita Mittel
NVIDIA
amittel@nvidia.com

Abstract

We propose a strategic approach to generate synthetic data in order to improve machine learning algorithms such as Deep Neural Networks (DNN). Utilization of synthetic data has shown promising results yet there are no specific rules or recipes on how to generate and cook synthetic data. We propose imitation training as a guideline of synthetic data generation to add more underrepresented entities and balance the data distribution for DNN to handle corner cases and resolve long tail problems. The proposed imitation training has a circular process with three main steps: First, the existing system is evaluated and failure cases such as false positive and false negative detections are sorted out; Secondly, synthetic data imitating such failure cases is created with domain randomization; Thirdly, we train a network with the existing data and the newly added synthetic data; We repeat these three steps until the evaluation metric converges. We validated the approach by experimenting on object detection in autonomous driving.

1. Introduction

Data is one of the most important parts for the development of machine learning algorithms such as Deep Neural Networks (DNN). It is often observed that better performances were achieved from data rather than DNN architectures. Data is valuable if it is well-balanced and representative across various categories and conditions; however, collection of such data is laborious and time consuming and cannot guarantee to be representative of each case. Also, the ground truth labels by human labelers are error prone and inconsistent across different labelers.

With faster and more efficient Graphics Processing Units (GPUs), simulators are able to create photo-realistic scenes in real time. As data created by simulators gets more difficult for a human to distinguish from real data, the data becomes more valuable for training machine learning modules. Simulators can create scenarios with different locations, backgrounds, light sources, times of day, and weather conditions. Therefore, the usage of synthetic data emerged and several approaches showed promising results[9][22][23]. However, there are, as far as we are aware of, no clear approaches on how and what synthetic data should be generated.

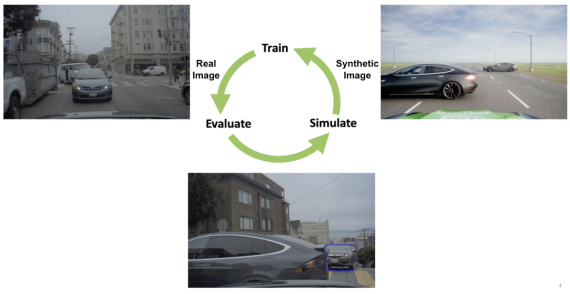

Figure 1: A cyclic flow of imitation training. After each evaluation of the existing model, failure cases are imitated by a simulator and then trained and aggregated in dataset.

We propose an efficient and effective synthetic data generation approach to improve performance of machine learning models using *imitation training*. First, we train a baseline model with real data. After evaluating this baseline machine learning model, scenes with incorrect predictions are collected. Then, those scenarios are recreated by a simulator, using the object's ground truth information, with certain perturbations. The machine learning models are trained again with newly created synthetic data as well as existing data. These three steps are repeated until the improvements are saturated. The flow is illustrated in Figure 1. This repeated imitation training requires that the validation and test data contains all representative objects, scenes and events.

In this paper, we mostly focus on the autonomous driving domain for DNN training. However, the approach can be

applied to other domains such as medical imagery, speech recognition, and with any machine learning algorithms utilizing simulation data.

2. Previous Works

In recent years, as simulated images have become more realistic, the popularity of synthetic datasets is rising as a reliable source of machine learning training and validation. There are multiple attempts to use synthetic data to improve their machine learning models such as Flying Chairs [5], FlyingThings3D [19], MPI Sintel [2], UnrealStereo [21], SceneNet [12], SceneNet RGB-D [20], SYNTHIA [23], GTA V [22], Sim4CV [18], Apollo [11] and Virtual KITTI [9].

Particularly, Alhaija et al. [1] discussed the use of simulated generated data to efficiently create datasets that can be used to create larger datasets. They focused on making realistic 3D Data for urban driving scenes. They augmented the KITTI dataset with realistic rendering. They were able to impose the synthetic data onto the real data in order to improve their training. Their pipeline consists of manual placement of objects in a scene and augmentation of the synthetic objects on a real background image. Adding these synthetic augmented objects increases the mAP by 3-4% from training only with real data although this was tested with a very small dataset of less than 4000 images.

Domain randomization is also relevant to our work as it is a useful tool to make robust datasets. Tremblay et al. [31] focused on using domain randomization with a simulator to create more data that can be used to collect data and avoid time consuming data collection and manual labeling. They investigated if using synthetic domain randomized data improves the results on real world data. In their domain randomization they mainly focused on adding objects like vehicles into the real world and randomizing the color and the texture. They tried multiple different networks including Faster R-CNN, R-FCN, SSD. They were able to get some improvement of the AP with added synthetic data.

Saleh et al. [26] used a game-engine based simulator, Grand Theft Auto to generate data. They generated synthetic data for the accurate semantic segmentation of autonomous driving scenes. After training their model with the synthetically generated data, they applied trained model to real world images and examined how adding these synthetic data will affect the results. They experimented with various networks and datasets and found out that adding higher quality data improved their results.

A common method of adding synthetic data to improve a deep neural network is by using transfer learning. Douarre et al. [7] used transfer learning with synthetic images in image segmentation in order to reduce the training time. They used a pre-trained CNN which was trained on similar real data and were able to achieve a higher quality measure. Transfer learning with synthetic data is a widely used technique [14][16][25] and we have more details in our experiments section.

Another technique to combine synthetic and real data is applying blur to the synthetic data. Vasiljevic et al. [32] experimented with mixing blurred and sharp images to improve the network. They found that adding blurred data improved the accuracy on blurry data as well as sharp data. Synthetic data is often much sharper than real data so blurring synthetic data should prove to make the data more realistic.

Imitation learning has become a popular method in the domains of robotics and motion planning. Codevilla et al. [4] used imitation learning, to learn control signals of autonomous driving. Driving behavior was learned from simulated environments and applied the learned lateral and longitudinal control signals into another simulated world. Schaal et al. [27] discussed model based imitation learning. Given knowledge of the task goal, the task-level policy of the movement primitive can be computed with reinforcement learning procedures based on the learned model. They showed effectiveness of imitation learning to make humanoid robots. Ho et al. [15] discussed a generative adversarial imitation learning algorithm and found that it worked better than Behavioral cloning, Feature expectation matching, and Game-theoretic apprenticeship learning. Duan et al. [8] discusses using one shot imitation learning to teach a robot how to perform a set of tasks by training the model with a subset of the tasks that the robot will be expected to complete.

Teacher-student learning is another effective domain adaptation approach as described by Li et al. [17]. The teacher-student network has two separate networks: The teacher network is complex and is trained with the entire dataset; The student network is less complex and tries to replicate the teacher network's outputs at each layer. Usually, the output from the teacher network (at different layers) can be used with the student network to get a convergence that works well for both networks. The main difference of the proposed imitation training from the teacher-student network is that unlike teacher-student network, imitation training keeps only a single network and focuses on creating synthetic data and aggregates it with existing datasets.

Online hard example mining is a reinforcement learning method that identifies difficult examples and augments the dataset with more of those difficult examples. Shrivastava et al [28] have found that by adding a few hard examples to an object detection dataset they were able to improve the overall mAP. This technique is similar to imitation learning as the model is augmented with "hard" cases where it has the lowest precision and accuracy.

3. Imitation Training

As discussed earlier, imitation learning is commonly used in robotics for an agent to learn policy from an expert. Contrary to conventional imitation learning, which learns the optimal policy for an agent, our goal is to train DNN to achieve better performance by imitation learning, which we call *imitation training*. Our imitation training approach can be seen as a hard-example mining strategy that uses synthetic imagery to generate difficult examples. However, in this strategy, Shrivastava et al [28] uses real data which is difficult to collect especially in dangerous situations (e.g. pedestrian in front of the vehicle). We chose to use imitation training as we found that many of the failure cases came from hard examples (e.g. heavy occlusion, truncation, etc.). Imitation training finds optimal policy π^* of synthetic data generation which has minimum expected loss inspired by reduction of imitation learning [24] as follows:

$$\pi^* = \arg\min_{\pi \in \prod} E_{d_\pi} \left[l\left(s, \pi\right) \right] \tag{1}$$

Where π is a policy to add new synthetic dataset, the state s is the current DNN and d_π is the average distribution of states over a cycle. l is a binary loss function with $l \in [0, 1]$ (0 if detection is true positive or true negatives, 1 if detection is false positive or false negative). Figure 1 shows a circular flow of imitation training with three steps, evaluate, simulate and train. Yue and Le [33] divided imitation learning into three categories, behavior cloning, inverse reinforcement learning and direct policy learning. Based on their categorization, our approach is between behavior cloning and direct policy learning.

3.1. Evaluation and Collection of Feedback

As a first step of the cycle, the existing network is evaluated, and the loss *l* is computed. The initial state would be the state of the model trained on real data before applying imitation training. The confidence threshold value for the network output is automatically selected to maximize the F1 score. With the threshold value we can collect all false positive(FP) and false negative(FN) cases in the validation dataset. The loss function *l* of current state is computed by the sum of the collected FP and FN.

3.2. Simulation (Imitator)

When there are M number of FP and FN frames, M sets of scenes will be created. Multiple FP and FN cases can happen in a frame and those cases will be simulated in one synthetic frame. We call an image frame as a scene in this paper. To create a synthetic scene, the 3D ground truth information is used. Since the data was collected and labeled using multiple lidars, radars, and cameras, it contains the 3D information of dynamic objects such as 3D location, orientation, dimension, type, etc. In order to avoid the network from overfitting the imitator creates a set of similar scenes with minor perturbations to the location and orientation. It also uses various weather conditions (sunny, cloud, fog, rain, snow), times of day, and backgrounds to further diversify the data. As shown by Figure 2 we are able to take scenes that have FN and/or FP and imitate that scene in the simulator to improve the model's accuracy. The backgrounds are different from the real data as we focus on imitating the object positions and various backgrounds acts as domain randomization to make the model more robust.

Figure 2: Imitation data generated with the simulator based off of the real data's 3D features. Adding perturbation to these imitated scenes is necessary as it prevents any chances that of overfitting our model to the test set.

3.3. Data Aggregation and Training

To let the DNN remember all previous mistakes, all of the training data is aggregated as the cycle repeats similar

Algorithm 1 Imitation Training

Input: False Positive (FP) and False Negative (FN) scenes with 3D ground truth information for objects (O). Number of times each scene is imitated (N). Current dataset D. Number of imitation cycles (C).

for $c = 1$ *to* C **do**
  $D_1 = \{\}$
  for *scene in* $\{FP, FN\}$ **do**
    for $i = 1$ *to* N **do**
      Randomize weather
      Randomize time of day
      Randomize background
      for *o in O in scene* **do**
        Obtain 3D information of object o
        Randomize object types within category
        Perturb 3D location of o
        Perturb 3D orientation of o
        Spawn object o in the simulator
      end
      Record the imitated scene s
      Accumulate scene s into dataset D_1
    end
  end
  Train the model with loss functions (3)(4) with $D + D_1$
  Evaluate the model using Equation (1)
  if *the number of false detection decreases* **then**
    $D = D + D_1$ using Equation (2)
  end
end

Output: Improved DNN Model

to [24]. At iteration time T, we aggregate newly created data D_T to the existing datasets D as shown below.

$$D = ((((D_0 \cup D_1) \cup D_2) \dots D_{T-1}) \cup D_T) \quad (2)$$

where D_0 is the real data in our experiment. $D_{1..T}$ are synthetically generated data.

Our object detection DNN model uses ResNet [13] blocks as the main feature extractor and outputs detection proposals from output grids that are finally clustered to form final detection results. Each layer of the model will generalize features of each object and using imitation training our goal is to improve the quality and accuracy of these generalizations. The network architecture is shown in Figure 3. For a loss function, binary cross entropy (BCE) in Equation(3) are used for coverage, occlusion and classification layers and L_1 loss in Equation (4) was used for the bounding box regression layer.

$$BCE = -\frac{1}{N}\sum_{i=1}^{N}\{y_i \cdot log(p(y_i)) + (1-y_i) \cdot log(1-p(y_i))\} \quad (3)$$

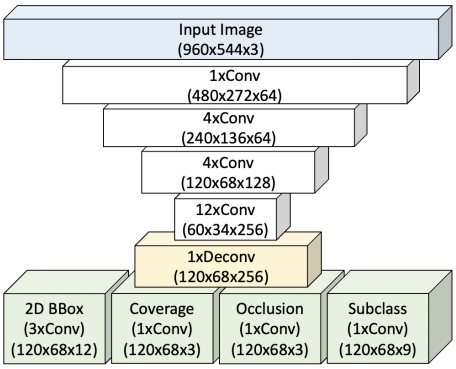

Figure 3: Architecture of our deep network based on ResNet

Where N is the output size, $p(y_i)$ is the i-th scalar value in the model output and y_i is the corresponding target value.

$$\text{L1 Loss} = \sum_{i=1}^{N} |y_{true} - y_{predicted}| \quad (4)$$

For more controlled imitation training, we could add newly generated data per scene one at a time to judge its effectiveness. However, training would take tremendous processing time to train each case sequentially. For efficient yet effective training, we add all cases of new synthetic scenes to one batch of data and inspect the performance for each case after training. A batch of synthetic dataset which does not improve the overall performance are rejected. The complete imitation training algorithm is laid out in Algorithm 1.

We repeat three steps of evaluation, simulation, and training until the improvement saturates or the portion of synthetic data reaches an upper limit. The convergence of data aggregation based imitation learning was proven at [24]. Experimentally with enough capacity of the neural network with deep layers and long epoch of training, addition of synthetic data did not deteriorate in performance of other cases. However, in practice, we do have an upper limit of synthetic data ratio (50%) with respect to real data since when the ratio of synthetic data is dominant, the network starts to overfit to synthetic data.

Nonetheless, the performance of the imitation training will saturate at a certain point when the simulator does not have the same or similar assets of the mis-detected objects (e.g. unusual road debris) or it cannot simulate a similar environment (e.g. snowy country roads).

4. Experiments

We conducted experiments of imitation training on an in-house dataset and publicly available datasets: Waymo Open Dataset [29] and KITTI dataset [10].

4.1. Simulator

In a choice of a simulator, there are multiple game-engine based simulators for autonomous driving scenario [6][23]. To train driving, planning, and control, any driving game would be fine. To train perception networks, the image quality of a simulator should be photo-realistic. Most widely used game engines are Unity [1], Unreal Engine[2], and GTA [22]. We used a simulator based on Unreal Engine. The simulator provides realistic rendering with diffused shadow, reflection of lights, water droplet on lens or windshield, diffused lights, the glaring sun. The simulator can reproduce any scenario with various weather conditions, locations and time for domain randomization [30]. The simulator can place obstacles (vehicles, pedestrians, bicycles, etc.) at specific locations or randomly. This is useful to recreate hard scenarios accurately.

4.2. In-House Dataset

Our in-house dataset consists of real-world highway, urban, and suburban data. This data is similar in nature to other autonomous driving datasets like: Waymo, KITTI, or Berkeley DeepDrive. The dataset is quite diverse as it has about 1.1 million images with varied weather (rain, sunny, cloudy, etc.), time of day (night, day, dawn, dusk), and backgrounds. The dataset also includes various corner cases and challenging scenes. Once the data was collected it was manually labeled and the quality was refined by multiple iterations. Since the dataset and the network are mature, it is extremely difficult to improve the performance.

4.2.1 Feature Comparison - Synthetic vs Real

Zagoruyko et al. [34] explains how a Convolutional Neural Network can compare subsections of an image and match different features. Imitation training is used to improve the feature matching of the real data. Although synthetic data is not 100% realistic, there are numerous common features between synthetic data and real data. For example, we can inspect discrete layers of the model to see how our model is trained and if it is able to detect the similar general features from the synthetic data. The model was trained on only real data and Figure 4 shows that even though the synthetic image is not as realistic as the real image the general features of the vehicles are quite similar each other. As long as the feature maps of real and synthetic data are close, the neural network will treat both images the same way. These feature maps in Figure 4 also show that synthetic data can be used for imitation training.

Through our tests we found that adding synthetic data that resembled False Positives and False Negatives in our real data improved the mean average precision of the model. We added this synthetic data that resembled false detections through imitation training.

Before applying imitation training, we experimented with a few other methods to combine real and synthetic data.

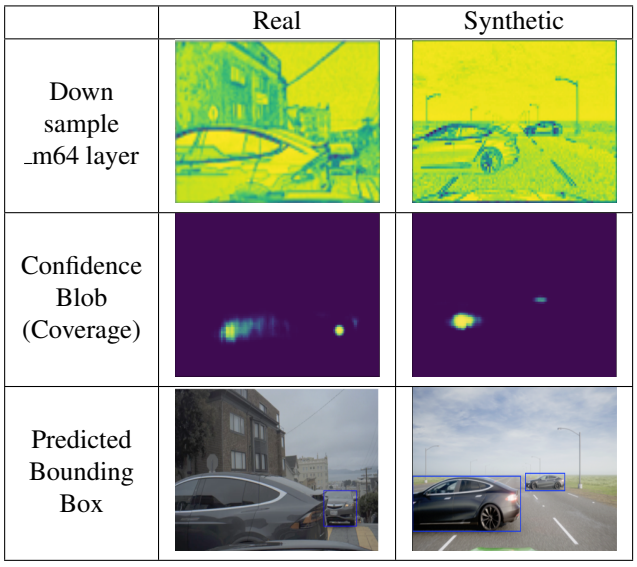

Figure 4: Comparison of real (left column) vs synthetic (right column) image feature maps of higher-level CNN layers. The feature maps are quite similar to each other and hard to be distinguished if the image is synthetic or real. The DNN model could not detect the truncated vehicle in the left real image as shown in the bottom left when trained on real data only. Therefore, the right synthetic image was generated for the DNN model to learn such features.

4.2.2 Dataset Ratio

As we progressed with our experiments, we began to focus on how the ratio of real to synthetic data affects the mean average precision and tried to determine the best ratio of data. In our initial experiments we found that sometimes adding more synthetic data would result in a reduction of mAP, so we began experimenting with different ratios to determine the best results. Some of the data we gathered is outlined in Figure 9. We found that the performance depended on a few factors. We found that increasing the ratio of synthetic data would improve Key Performance Indicator(KPI). However, as we further increased the amount of synthetic data, we started to face some regression. Both of these strategies were moderately successful and afterwards we started developing imitation training which we will discuss in the next section.

[1] https://unity.com
[2] https://www.unrealengine.com/en-US/

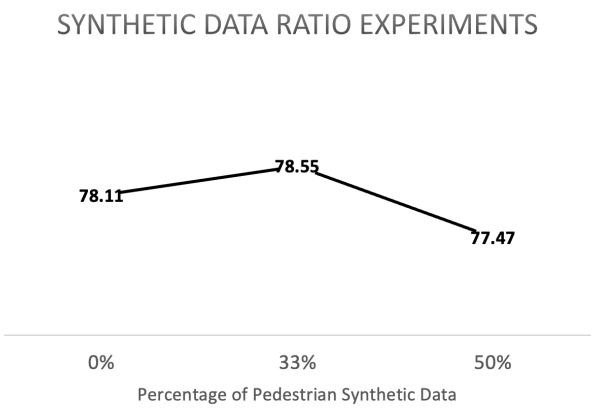

Figure 5: Experiments on different ratios of real and synthetic data.

4.2.3 Transfer Learning

To take advantage of synthetic data to improve our models we tried using a pretrained model. The model was trained solely on real data and using transfer learning we froze layers in the model and retrained with synthetic and real data. The goal was to extend the real model using similar but different data. We tried freezing the model at different points and found that the optimal layers to freeze was from the first to 245-th layer. We found that freezing the layers as a transfer learning technique performs better than the baseline method but it was still not the best solution to using synthetic data.

KPI: mAP(%)		**Test**			
		Real		Sim	
Train	Real (Baseline)	Car	92.39	Car	89.62
		Bike	75.90	Bike	63.40
		Person	78.11	Person	49.23
	Freeze 243 layers	Car	92.21	Car	**89.65**
		Bike	74.08	Bike	**79.92**
		Person	77.99	Person	**73.90**
	Real + Sim Freeze 245 layers	Car	**92.25**	Car	89.56
		Bike	74.35	Bike	78.31
		Person	**77.88**	Person	73.00
	Real + Sim Freeze 249 layers	Car	92.24	Car	89.56
		Bike	**74.44**	Bike	78.31
		Person	77.87	Person	73.16

Table 1: Experiments on transfer learning by freezing layers.

4.2.4 Imitation Training Experiments

In the last row of Figure 4, there was a false negative detection on a partially visible Tesla. Therefore, multiple similar synthetic images were generated using a game-engine based simulator. An example of synthetically generated images is shown at the bottom right of Figure 4. Such synthetic data was added in our training dataset and the new network was trained with other real and synthetic data. The synthetic data has randomized features (e.g. background, weather, time of day, etc.) may increase the overall effectiveness of the model. The newly trained model was evaluated on an evaluation dataset and the cases of false positive and false negative detections were reported and the cycle was repeated.

The other example is shown in Figure 6 & Figure 7. The existing DNN trained on real data had an issue in detecting partially visible trucks and occluded vehicles as shown in Figure 6. Therefore, the imitator created multiple scenes with partially visible trucks at various places, times of day, and weather conditions for domain randomization. The example of synthetically generated images imitating Figure 6 are shown in Figure 7. Along with other synthetic data, the network was retrained. After adding the synthetic data, the new network could detect partially visible trucks correctly as shown in Figure 6.

Figure 6: Left: Incorrect detection (yellow rectangle) on a truck and occluded vehicle using a model trained on real data. The gray rectangle indicates ground truth and blue rectangle indicates predicted detection. Vehicles with gray rectangle only indicate false negative detections. Right: Correct detection after imitation training using the synthetic data.

Train with synthetic data only: The second row of Table 2 shows the performance of a network trained on synthetic training data only. It performed the worst with real test data but did fine with synthetic data. This was an expected result because the synthetic data alone could not represent highly complicated real world scenes.

Train with real and synthetic data without imitation training: The third row of Table 2 shows the performance of the network trained on real data and randomly generated synthetic data. The datasets we generated were not based

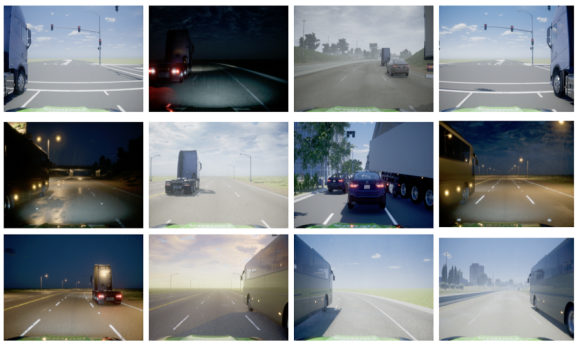

Figure 7: Examples of synthetically generated images imitating the false truck detection in the upper images in Figure 6.

on the false detections from the baseline model. The results were better than training on only synthetic data but there was significant regression from the baseline model.

Train with real and synthetic data using imitation training: The third to sixth rows of Table 2 indicate the imitation training process. In the first cycle in the third row the new network performed worse than the baseline network. However, as we add more data and capture more corner cases and long tail cases, the performance improved over time and reached the best performance as compared with the baseline (which was only trained with real data) at the fourth cycle. The number of cycles needed to get an improvement is arbitrary and for this dataset it took four imitation cycles.

KPI: mAP(%)		**Test**			
		Real		Sim	
Train	Real (Baseline)	Car	92.39	Car	89.62
		Bike	75.30	Bike	63.40
		Person	78.11	Person	49.23
	Sim	Car	40.07	Car	93.34
		Bike	3.43	Bike	89.21
		Person	3.81	Person	93.17
	Real + Sim (No Imitation)	Car	89.63	Car	90.09
		Bike	72.72	Bike	73.93
		Person	76.68	Person	76.96
	Real + Sim (Cycle 1)	Car	92.38	Car	92.25
		Bike	75.15	Bike	93.26
		Person	78.17	Person	94.40
	Real + Sim (Cycle 2)	Car	**92.56**	Car	**95.63**
		Bike	75.14	Bike	**93.65**
		Person	**78.77**	Person	94.70
	Real + Sim (Cycle *N*)	Car	92.50	Car	95.62
		Bike	**75.63**	Bike	93.01
		Person	78.47	Person	**94.80**

Table 2: Cross-validation of training and evaluation for each imitation training cycle.

4.3. Waymo Open Dataset

Table 3 shows the results of imitation training on Waymo's Open Dataset with synthetic data. We trained the in-house object detection DNN model illustrated in Figure 3 with Waymo data. For training, we used Waymo's training dataset and synthetic imitation data. For test, we used both Waymo's test dataset and the synthetic test dataset. The results clearly show that adding synthetic imitation data improves the model even with different dataset.

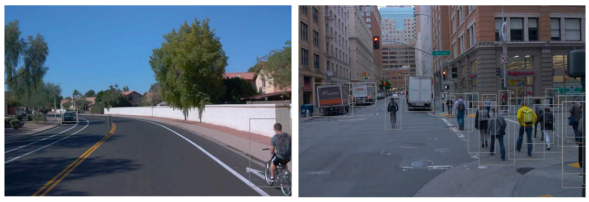

Figure 8: Waymo dataset. The grey bounding boxes of a cyclist and persons are quite large, which caused low mAP for both classes.

4.3.1 Image Augmentation

In order to make synthetic data more realistic we focused on utilizing different image augmentations. One image augmentation we found particularly useful was a Gaussian Blur. We trained the Waymo Open Data set with synthetic imitation data and applied a 5×5 Gaussian Blur kernel on the synthetic data in order to reduce the sharpness in synthetic images. At Table 3 we can compare the results from the second and third rows to the baseline and we see that the results where the synthetic data was blurred improved mAP with a significant margin. This improvement comes from the Gaussian Blur that made the synthetic data image histograms more similar to the real data image histograms.

KPI: mAP(%)		**Test**			
		Waymo		Sim	
Train	Waymo	Car	73.79	Car	77.95
		Bike	25.19	Bike	18.55
		Person	46.04	Person	8.31
	Waymo + Synthetic	Car	74.12	Car	**88.83**
		Bike	24.98	Bike	92.48
		Person	47.45	Person	**93.13**
	Waymo + Synthetic Blur	Car	**74.58**	Car	88.01
		Bike	**28.13**	Bike	**92.79**
		Person	**47.06**	Person	92.56

Table 3: Ablation test with cross-validation of training and evaluation for Waymo Open Dataset and synthetic data. The blur was applied only on synthetic images.

4.4. KITTI Dataset

We also applied imitation training on the KITTI dataset with a state-of-the-art vehicle detection model, SimpleDet [3]. We trained a SimpleDet model with real KITTI data and used imitation training to generate synthetic data. There were 16,711 images of real KITTI data and we generated 15,366 images of imitation data. The synthetic data imitated the frames of false detections mostly in distant, truncated, or occluded objects in the KITTI data. SimpleDet using faster-RCNN as a backbone had the best performance in Waymo's 2D Object Detection Challenge. We trained the model from scratch with KITTI and imitated synthetic data and tested on KITTI test data. Table 4 clearly shows that the imitation training with KITTI + Sim outperformed the the state-of-the-art model in both precision and recall. Figure 9 shows a comparison against other 2D detection methods on the KITTI test set. SimpleDet with synthetic imitation data giving the best performance in vehicle detection.

KPI: mAP(%)		**Test**	
		KITTI	
Train	KITTI	Mean Average Precision	67.9
		Mean Average Recall	72.5
	KITTI + Sim	Mean Average Precision	**68.2**
		Mean Average Recall	**72.8**

Table 4: Ablation test with cross-validation of training and evaluation for real KITTI dataset and synthetic data. We used SimpleDet for training and evaluation. The mAP (Mean Average Precision) is the average precision over every IoU threshold. Likewise, the mAR (Mean Average Recall) is the average recall over every IoU threshold.

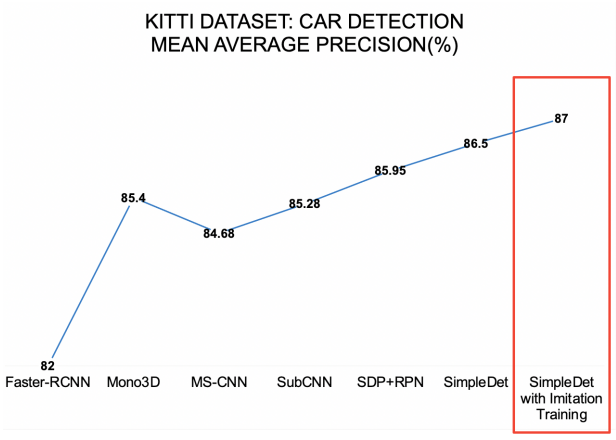

Figure 9: Performance comparison of recently published works and using imitation training with SimpleDet. Each metric is the average precision on the corresponding class.

5. Conclusion

Through this paper we discussed the benefits of using synthetic data and introduced imitation training to guide the synthetic data generation and improve the performance of DNNs. We found that adding synthetic imitation data to a DNN by imitating false detections resulted in a significant increase in the mAP on real and synthetic data. This reduces the need for extensive data gathering and labeling and shows that using synthetic data is a viable method of improving a DNN. We validated our approach on our object detection for autonomous vehicles. Imitation training may be extended to segmentation, depth estimation, classification and other applications.

References

[1] Hassan Abu Alhaija, Siva Karthik Mustikovela, Lars Mescheder, Andreas Geiger, and Carsten Rother. Augmented reality meets computer vision: Efficient data generation for urban driving scenes. *International Journal of Computer Vision*, 126(9):961–972, 2018. 2

[2] Daniel J Butler, Jonas Wulff, Garrett B Stanley, and Michael J Black. A naturalistic open source movie for optical flow evaluation. In *European conference on computer vision*, pages 611–625. Springer, 2012. 2

[3] Yuntao Chen, Chenxia Han, Yanghao Li, Zehao Huang, Yi Jiang, Naiyan Wang, and Zhaoxiang Zhang. Simpledet: A simple and versatile distributed framework for object detection and instance recognition. *Journal of Machine Learning Research*, 20(156):1–8, 2019. 8

[4] Felipe Codevilla, MüMatthias ller, LóAntonio pez, Vladlen Koltun, and Alexey Dosovitskiy. End-to-end driving via conditional imitation learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4693–4700. IEEE, 2018. 2

[5] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Van Der Patrick Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2758–2766, 2015. 2

[6] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. In *Conference on robot learning*, pages 1–16. PMLR, 2017. 5

[7] Cl Douarreément, Richard Schielein, Carole Frindel, Stefan Gerth, and David Rousseau. Transfer learning from synthetic data applied to soil–root segmentation in x-ray tomography images. *Journal of Imaging*, 4(5):65, 2018. 2

[8] Yan Duan, Marcin Andrychowicz, Bradly C Stadie, Jonathan Ho, Jonas Schneider, Ilya Sutskever, Pieter Abbeel, and Wojciech Zaremba. One-shot imitation learning. *arXiv preprint arXiv:1703.07326*, 2017. 2

[9] 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. 1, 2

[10] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The KITTI dataset. *International Journal of Robotics Research*, 32(11):1231 – 1237, Sept. 2013. 4

[11] Yuliang Guo, Guang Chen, Peitao Zhao, Weide Zhang, Jinghao Miao, Jingao Wang, and Tae Eun Choe. Gen-lanenet: A generalized and scalable approach for 3d lane detection. In *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XXI*, volume 12366 of *Lecture Notes in Computer Science*, pages 666–681. Springer, 2020. 2

[12] Ankur Handa, Viorica Patraucean, Vijay Badrinarayanan, Simon Stent, and Roberto Cipolla. Understanding real world indoor scenes with synthetic data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4077–4085, 2016. 2

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 4

[14] Tobias Heimann, Peter Mountney, Matthias John, and Razvan Ionasec. Real-time ultrasound transducer localization in fluoroscopy images by transfer learning from synthetic training data. *Medical image analysis*, 18(8):1320–1328, 2014. 2

[15] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *arXiv preprint arXiv:1606.03476*, 2016. 2

[16] Seokwon Jung, Jungbae Park, and Sangwan Lee. Polyphonic sound event detection using convolutional bidirectional lstm and synthetic data-based transfer learning. In *ICASSP 2019-2019 IEEE International Conference on Speech Acoustics and Signal Processing (ICASSP)*, pages 885–889. IEEE, 2019. 2

[17] Jinyu Li, Michael L Seltzer, Xi Wang, Rui Zhao, and Yifan Gong. Large-scale domain adaptation via teacher-student learning. *arXiv preprint arXiv:1708.05466*, 2017. 2

[18] MüMatthias ller, Vincent Casser, Jean Lahoud, Neil Smith, and Bernard Ghanem. Sim4cv: A photo-realistic simulator for computer vision applications. *International Journal of Computer Vision*, 126(9):902–919, 2018. 2

[19] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for optical flow and disparity scene flow estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4040–4048, 2016. 2

[20] John McCormac, Ankur Handa, Stefan Leutenegger, and Andrew J Davison. Scenenet rgb-d: 5m photorealistic images of synthetic indoor trajectories with ground truth. *arXiv preprint arXiv:1612.05079*, 2016. 2

[21] Weichao Qiu and Alan Yuille. Unrealcv: Connecting computer vision to unreal engine. In *European Conference on Computer Vision*, pages 909–916. Springer, 2016. 2

[22] Stephan R Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In *European conference on computer vision*, pages 102–118. Springer, 2016. 1, 2, 5

[23] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio M Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3234–3243, 2016. 1, 2, 5

[24] St Rosséphane, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011. 3, 4

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

[26] 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

[27] Stefan Schaal. Is imitation learning the route to humanoid robots? *Trends in cognitive sciences*, 3(6):233–242, 1999. 2

[28] Abhinav Shrivastava, Abhinav Gupta, and Ross Girshick. Training region-based object detectors with online hard example mining. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 761–769, 2016. 2, 3

[29] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2446–2454, 2020. 4

[30] Jonathan Tremblay, Aayush Prakash, David Acuna, Mark Brophy, Varun Jampani, Cem Anil, Thang To, Eric Cameracci, Shaad Boochoon, and Stan Birchfield. Training deep networks with synthetic data: Bridging the reality gap by domain randomization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 969–977, 2018. 5

[31] Jonathan Tremblay, Thang To, Balakumar Sundaralingam, Yu Xiang, Dieter Fox, and Stan Birchfield. Deep object pose estimation for semantic robotic grasping of household objects. *arXiv preprint arXiv:1809.10790*, 2018. 2

[32] Igor Vasiljevic, Ayan Chakrabarti, and Gregory Shakhnarovich. Examining the impact of blur on recognition by convolutional networks. *arXiv preprint arXiv:1611.05760*, 2016. 2

[33] Yisong Yue and Hoang Le. Imitation learning tutorial. *Tutorial at ICML*, 2018, 2018. 3

[34] Sergey Zagoruyko and Nikos Komodakis. Learning to compare image patches via convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4353–4361, 2015. 5