Driving in the Matrix: Can Virtual Worlds Replace Human-Generated Annotations for Real World Tasks?

Matthew Johnson-Roberson[1], Charles Barto[2], Rounak Mehta[3], Sharath Nittur Sridhar[2], Karl Rosaen[2], and Ram Vasudevan[4]

***Abstract*— Deep learning has rapidly transformed the state of the art algorithms used to address a variety of problems in computer vision and robotics. These breakthroughs have relied upon massive amounts of human annotated training data. This time consuming process has begun impeding the progress of these deep learning efforts. This paper describes a method to incorporate photo-realistic computer images from a simulation engine to rapidly generate annotated data that can be used for the training of machine learning algorithms. We demonstrate that a state of the art architecture, which is trained *only* using these synthetic annotations, performs better than the identical architecture trained on human annotated real-world data, when tested on the KITTI data set for vehicle detection. By training machine learning algorithms on a rich virtual world, real objects in real scenes can be learned and classified using synthetic data. This approach offers the possibility of accelerating deep learning's application to sensor-based classification problems like those that appear in self-driving cars. The source code and data to train and validate the networks described in this paper are made available for researchers.**

***Index Terms*— deep learning, simulation, object detection, autonomous driving**

I. Introduction

The increasingly pervasive application of machine learning for robotics has led to a growing need for annotation. Data has proven to be both the limiting factor and the driver of advances in computer vision, particularly within semantic scene understanding and object detection [1], [2]. A variety of approaches have been proposed to efficiently label large amounts of training data including crowdsourcing, gamification, semi-supervised labeling, and Mechanical Turk [3]–[5]. These approaches remain fundamentally bounded by the amount of human effort required for manual labeling or supervision.

To devise a strategy to address this ever-growing problem, this paper proposes the use of computer annotated photo-realistic simulations to train deep networks, enabling robots to semantically understand their surroundings. In particular, we focus on the task of vehicle detection within images. Although this application is specialized, the insights developed here can cut across a variety of perception-related tasks in robotics. This vehicle detection task also serves as a useful motivator since a human driver's ability to operate safely using only visual data illustrates there is sufficient signal in images alone to safely operate an automobile. The goal of vision-only driving has become something of a Holy Grail in the autonomous vehicle community due to the low cost of sensors.

Currently this remains beyond the state-of-the-art of robotics, but several paradigms have been proposed. These lie on a spectrum from component-based systems, where combinations of codified rules and learned classifiers are applied to each of the individual sub-tasks involved in a complex action like driving, to end-to-end learning, in which sensor data is mapped directly to the action space of driving through learning from many examples. Deep-learning has become an extremely popular approach to solve many of the sub-problems in the componentized method [6], [7] and has even been proposed as the most plausible method to solve the end-to-end problem in self-driving cars if such a solution is even plausible [8].

Despite the potential to address a variety of problems in the self-driving car space, deep learning, like many purely data-driven approaches, is prone to overfitting. In fact, given the large number of parameters in deep learning systems, they have proven particularly prone to data set bias [9]–[11]. While data sets have spurred impressive progress in the research community, the lack of generalizability remains an impediment to wide adoption and deployment on fielded robotic systems [9]. In fact, this paper illustrates that current successful approaches to the vehicle detection task are based on a handful of data sets and a training process that leads to poor cross data set performance, which restricts real-world utility.

On the other hand, one can ask whether the relatively finite (order of thousands of images) current hand-labeled data sets are sufficiently large and diverse enough to ever learn general models that enable vehicles to operate safely in the world? To address this problem, we consider the same cross data set validation task, where a model is learned on a completely independent data set (i.e. different geographic regions, time of day, camera system, etc.) and tested on another. We propose and explore visual simulation as a solution to the issue of finite human labeled data sets.

Computer graphics has greatly matured in the last 30 years and a robust market for entertainment, including CGI movies, video games, and virtual reality, has spurred great

[1]M. Johnson-Roberson is with the Department of Naval Architecture and Marine Engineering, University of Michigan, Ann Arbor, MI 48109 USA mattjr@umich.edu

[2] C. Barto, S. Sridhar, and K. Rosaen are with the Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109 USA bartoc,sharatns,krosaen@umich.edu

[3] R. Mehta is with the Robotics Program, University of Michigan, Ann Arbor, MI 48109 USA rounak@umich.edu

[4] R. Vasudevan is with the Mechanical Engineering, University of Michigan, Ann Arbor, MI 48109 USA ramv@umich.edu

innovation in our ability to generate photo-realistic imagery at high speed. If networks trained with this type of synthetic annotated data can perform as well as networks trained with human annotated data on real-world classification tasks, then one can begin rapidly increasing the size of useful training data sets via simulation and verifying whether larger data sets are sufficient to learn general models.

The contributions of this paper are as follows: 1) a fully automated system to extract training data from a sophisticated simulation engine; 2) experiments to address the issue of data set bias in deep learning-based object classification approaches, specifically in self-driving car data sets; 3) state-of-the-art performance on real data using *simulation only* training images; and 4) results that highlight improved performance with greater numbers of training images, suggesting the ceiling for training examples has not yet been reached for standard deep learning network architectures.

The paper is organized as follows: Section II presents prior work; Section III describes the process used to collect the data and the training process; Section IV discusses the experimental setup; Section V presents preliminary results; Section VI discusses these preliminary results in detail; and finally, Section VII presents our conclusions and future work.

II. Related Work

There have been several attempts to generate virtual data to boost the performance of deep learning networks. Early approaches used 3D renderings of objects to augment existing training data for pedestrian detection task [12], [13]. Further work inserted synthetic pedestrians at a variety of angles into real scenes to train a part-based model classifier [14]. Su, Qi, Li, *et al.* [15] employed 3D rendering to perform synthetic image generation for viewpoint estimation to construct training data for a convolutional neural network (CNN) based approach.

More recently, fully synthetic worlds have been suggested. Most relevant to this paper is the work of Richter, Vineet, Roth, *et al.* [16], where the same engine we propose was used to generate simulated data for semantic segmentation of images. However, this work still required human annotators in the loop to supervise the generation of labels. Additionally, these data were not used alone but in concert with the real-world CamVid data set [17]. A network trained on the combined dataset achieved superior performance to the same network trained on the CamVid training set alone, as evaluated on the CamVid testing dataset.

Each of these approaches refine the performance of their approach by fine tuning networks using a portion of the real-world testing data. This can be explained by a common problem in machine learning where networks trained on one data set suffer in terms of performance when evaluated on another data set [9]. This is of significant concern in the context of autonomous vehicles given that these algorithms need to run in different areas of the world and in a range of weather and lighting conditions.

An important exception to this training and evaluation approach is the recent semantic image-based segmentation result constructed using the SYNTHIA data set [18], which trained on 13,000 purely synthetic images. However, the authors relied upon a mixture of real and synthetic images to train a network that achieved comparable performance to a network trained only on real-world data.

III. Technical Approach

This section describes our approach to generate synthetic images with computer-generated annotations of vehicles, which are then used to train our object detection network.

A. Cloud-Cased Simulation Capture

We endeavour to leverage the rich virtual worlds created for major video games to simulate the real world with a high level of fidelity. Specifically, we leverage Grand Theft Auto V (GTA V). The publisher of GTA V allows non-commercial use of footage from the game [16]. Data is captured from the game using two plugins, Script Hook V and Script Hook V.NET, developed by the open source community [19]. We refer to the first as the *native plugin* and the second as the *managed plugin*.

Our process to generate simulated data is as follows: The managed plugin captures information about the scene at 1 Hz and uploads it to a cloud machine running an SQL server. In addition, it retrieves screen shots, scene depth, and auxiliary information from the game stored in the graphics processing unit (GPU)'s stencil buffer data. Sample images appear in Fig. 1. As illustrated in Fig. 2, for each simulation capture point we can save a maximum of five images using different weather types.

Screen shots are captured by "hooking" into Direct3D 11's present callback. A process by which the native call is replaced with custom code which operates and then returns to the native call. In our custom code, we copy the graphics card's buffers. Specifically the depth and stencil data are captured by hooking into Direct3D's ID3D11ImmediateContext::ClearDepthStencilView and saving the buffers before each call. Because of optimizations applied by the graphics card drivers, we "rehook" the clear function each frame. When saving each sample the managed plugin requests all current buffers from the native plugin and the buffers are downloaded from the GPU and copied into managed memory. Buffers are saved in tiff format to zip archives and uploaded to a cloud server.

B. Internal Engine Buffers

The format of the rendering engine's stencil and depth buffers are somewhat atypical. The value stored in the depth buffer is actually the logarithm of the actual depth. This enhances depth precision far away from the camera, and is one of the techniques that allows the engine to render such a large world. Since we are interested in depth ranges and not just depth ordering, we linearize the depth buffer before using it in post processing. The stencil buffer is an 8-bit per pixel buffer used to store information about each

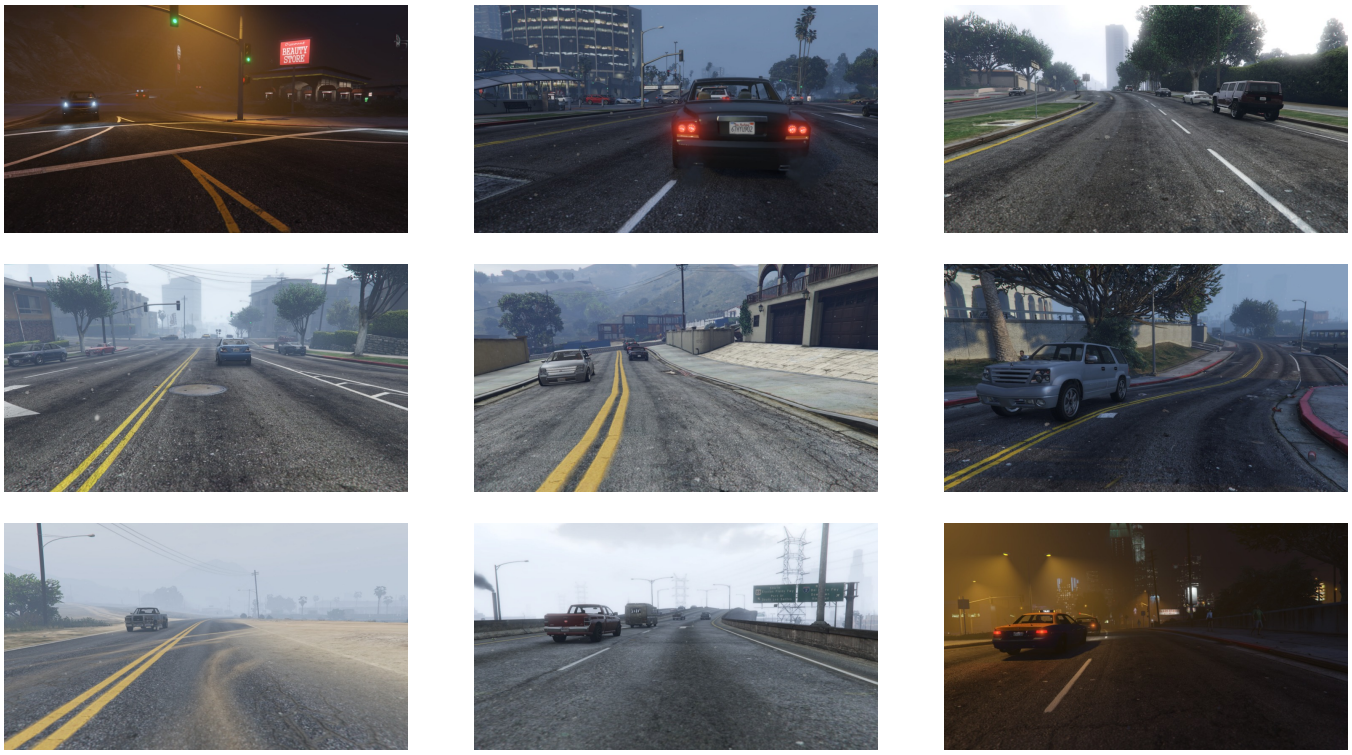


Fig. 1: Sample images captured from the video game based simulation engine proposed in this paper. A range of different times of day are simulated including day, night, morning and dusk. Additionally the engine captures complex weather and lighting scenarios such as driving into the sun, fog, rain and haze.

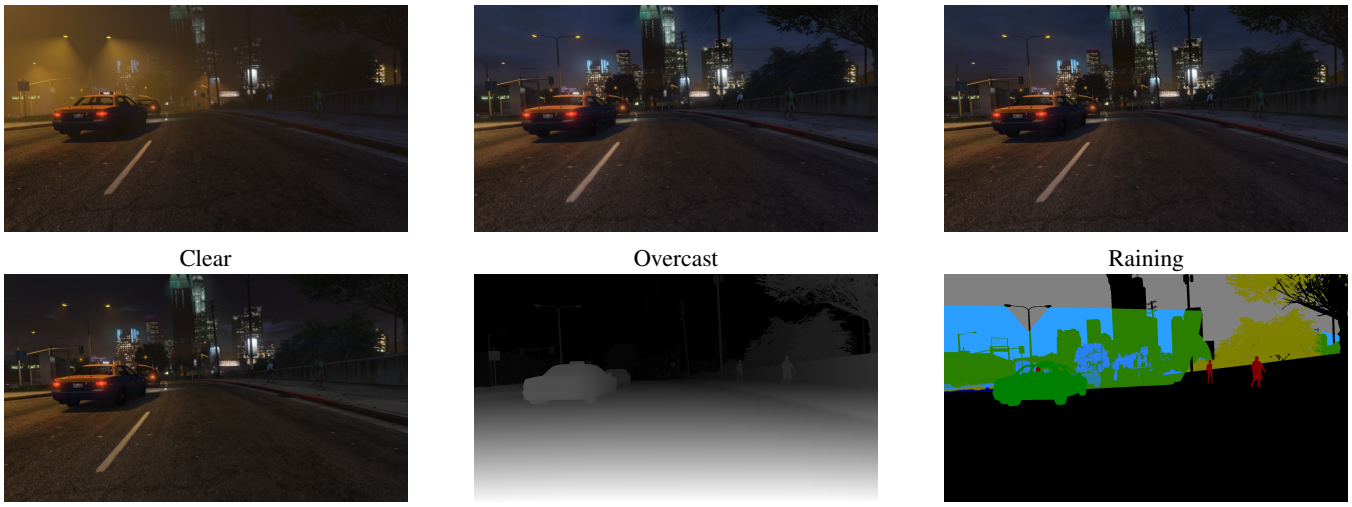


Fig. 2: Four different weather types appear in the training data. The simulation can be paused and the weather condition can be varied. Additionally note the depth buffer and object stencil buffer used for annotation capture. In the depth image, the darker the intensity the farther the objects range from the camera. In the stencil buffer, we have artificially applied colors to the image's discrete values which correspond to different object labels for the game. Note that these values cannot be used directly. The process by which these are interpreted is highlighted in Section III-B.

of the classes within a scene. It is utilized in the engine to store information about object class to answer queries such as "which object is under the cross hair." The engine splits its stencil buffer in half, using the bottom four bits to store a numerical object ID, and the top four bits to store some flags.

The managed plugin captures the 2D projection of each object's oriented bounding box, the position of each object, the position of the camera, and the object class. The projected bounding box, which we call the ground truth bounding box, is often loose so we post-process each ground truth detection using data from the stencil and depth buffers.

C. Tight Bounding Box Creation

The process to generate tight object bounding boxes without any human intervention is depicted at a high level in Fig. 3. The steps are as follows:

1) The engine mantains coarse bounding boxes from the 3D positions of all objects in a fixed radius from the player. However these may be too loose to be directly useful for training (see Figure 3a).
2) To refine these boxes, contour detection is run on the stencil buffer (shown in green). As the stencil buffer contains simple pixel class labels, detections that partially occlude each other generate a single contour. This problem can be seen in Fig. 3b.
3) Using the depth buffer we can compute the mean depth within the detected contour. See Fig. 3c and Fig. 3f which shows depths around the mean for the compact car.
4) We then generate a new putative bounding box where included pixels are thresholded based upon a distance from the mean depth calculated in the previous step. The contours of accepted pixels are shown in Figs. 3d – 3f. Notice how the contours for the truck and the car are now distinct. We found that this improves the quality of the ground truth bounding boxes significantly.
5) Finally we also add bounding boxes of pixels classified as "car" in the stencil buffer that still do not have a corresponding ground truth bounding box. The blue bounding box in Fig. 3d is an example of one such detection. This allows us to include cars that are too far away to be registered with the physics engine but are still rendered.

D. Dataset Properties

To train the learning architecture described in the next subsection, we generated 3 distinct simulated data sets of varying size: Sim 10k with $10,000$ images, Sim 50k with $50,000$ images, and Sim 200k with $200,000$ images. The size of these data sets is compared to the state of the art human annotated data sets of real-world scenes in Table I. A heatmap visualizing the distribution of the centroids of the car bounding boxes for the real-world scenes and the simulated scenes is illustrated in Fig. 4. Notice the much larger spread of occurrence location in the simulated data than in the real world data set. Additionally Fig. 5 depicts the distribution of the number of detections per frame. Notice that KITTI and the simulation data set are similarly distributed in terms of the number of detections per frame.

Data set	# of images
Cityscapes [20]	2,975
KITTI [6]	7,481
Sim 10k	10,000
Sim 50k	50,000
Sim 200k	200,000

TABLE I: This table presents the sizes of the data sets used for training and validation. Since one of the primary focuses of this paper is to understand data set bias in automated detection of objects for autonomous driving using vision, the performance of the network is evaluated on a data set that is distinct from the data set used to train the network. Note the relatively small sizes of the major real-world data sets used for training.

E. Object Detection Network Architecture

Since a specific network architecture is not the focus of this paper, we employ a state of the art deep learning object detection network architecture: Faster-RCNN implemented in the Mx-Net framework [21], [22]. Full details of the approach appear in the paper, however the following important departures from the original reference implementation were made: 1) use of VGG-16 [2] as opposed to AlexNet used in the original paper [21].

The training and evaluation procedures are standard for an end-to-end training of Faster-RCNN. Layers copied from the VGG-16 network were initialized with Imagenet [1] pre-trained weights in all cases.

IV. Experimental Design

All the network performance assessments are calculated by testing on real images from the KITTI data set. All 7481 images in the KITTI "training" set were used as the testing data for the proposed networks trained on any of the simulation or Cityscapes data sets. Since each data set varies in size so dramatically, there was no fixed number of total iterations that all of the networks could be trained on to achieve optimal performance. Bearing this in mind, each network was run until performance asymptoted. Using this criteria, the network trained on Cityscapes was run for 13 epochs, Sim 10k for 12 epochs, Sim 50k for 12 epochs and Sim 200k 8 epochs. Note that an epoch represents one complete run through the entire data set. Training began with a learning rate of 10^{-3} and decreased by a factor of 10 after every $10,000$ iterations until a minimum learning rate of 10^{-8} was achieved. We used a momentum factor of 0.9 and a weight decay factor of 5×10^{-4} along with a batch size of one image per GPU for all experiments.

The performance of each of the trained networks was evaluated using a standard intersection over union (IoU) criteria for produced bounding boxes [6]. We use a 0.7 IoU overlap threshold as is typical in KITTI evaluation on cars. In training the network on the Cityscapes data, we used rectangular bounding boxes that were obtained by taking the smallest bounding box that fully enclosed the "coarse" pixel annotations provided in the data set. Detections which are fully visible while having a minimum bounding box height of 40 pixels and a maximum truncation of 15% are

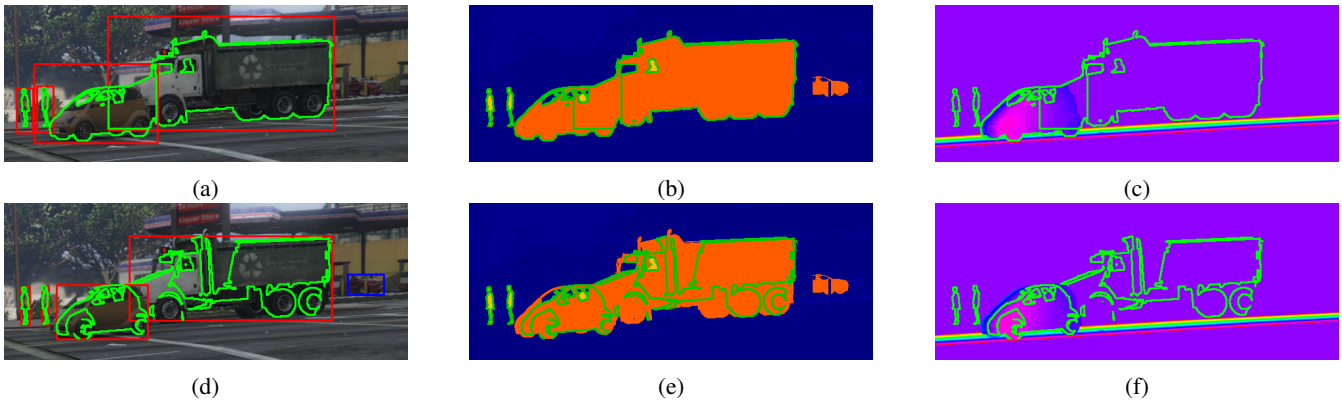


Fig. 3: An illustration of the pipeline for tight bounding box creation. The engine's original bounding boxes are shown in (a). Since they are loose, we process them before using them as training data. In (b) we see an image from the stencil buffer, the orange pixels have been marked as vehicle enabling us to produce tight contours outlined in green. However, note that the two objects do not receive independent IDs in this pass so we must disambiguate the pixels from the truck and the compact car in a subsequent step. To do this we use the depth shown in (c) where lighter purple indicates closer range. This map is used to help separate the two vehicles, where (e) contains updated contours after processing using depth and (f) contains those same updated contours in the depth frame. Finally, (d) depicts the bounding boxes with the additional small vehicle detections in blue which are all used for training. Full details of the process appear in Section III-C.

categorized as Easy. Detections which are partially occluded while having a minimum bounding box height of 25 pixels with a maximum truncation of 30% are categorized as Moderate. Detections which are difficult to see with a maximum truncation of 50% and a minimum bounding box height of 25 pixels are categorized to be Hard.

V. Results

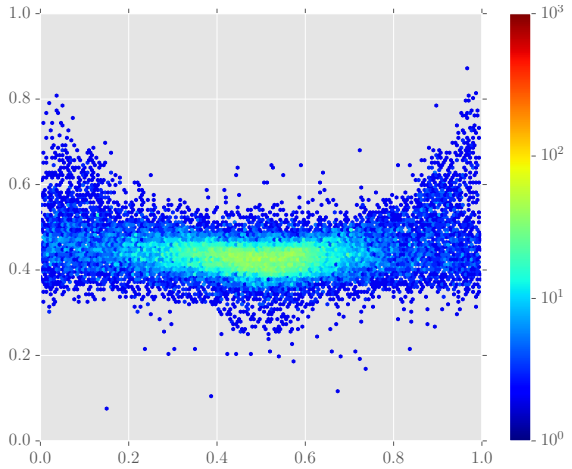
The goal of this work is to highlight the power of simulation and 3D engines as tools for the training of deep learning approaches for object classification. To this end Fig. 6 depicts the results of applying a typical Faster R-CNN (see Section III-E) network trained on 10,000, 50,000 or 200,000 simulation images. First, it is important to notice that the base number of simulation images used is much larger than the real image data sets (KITTI and Cityscapes). It appears the variation and training value of a single simulation image is lower than that of a single real image. The lighting, color, and texture variation in the real world is greater than that of our simulation and as such many more simulation images are required to achieve reasonable performance. However, the generation of simulation images requires only computational resources and most importantly does not require human labeling effort. Once the cloud infrastructure is in place, an arbitrary volume of images can be generated. Second, there is a significant jump between 10,000 and 50,000 images. This indicates to us that there is a threshold of images above which the network learns a much more discriminative model of cars in the world. Perhaps most intriguing, we see continued improved performance between 50,000 and 200,000 images (albeit a much smaller jump). To confirm this is not simply a function of higher iteration count we ran 50,000 images through additional training epochs and actually saw a decrease in performance (most likely due to overfitting to that specific data).

In Fig. 7, we see the qualitative results of increasing the number of training images in the simulation data set. Many smaller arguably more challenging cars that appear at greater distance in the scene are detected by the simulation networks trained on more images. The continued improvement with additional simulation images points to an interesting discussion point: perhaps we are limited by data set size as opposed to network architecture.

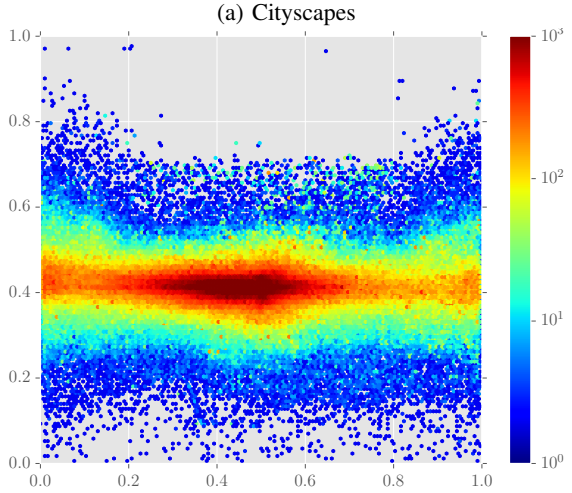
Beyond understanding the relative performance of varying numbers of simulation images it is important to understand how the simulation results relate to the performance on a network trained with annotated real images. Table II shows the results of both simulated trained networks, but also a network trained on real images drawn from a distinct data set (in this case Cityscapes). Notice that with larger number of only simulation images, we achieve superior performance to a network trained with only real images. In the table it can be seen that the Sim 50k and 200k datasets clearly outperform Cityscapes in the Easy, Moderate and Hard categories. Additionally, in Fig. 8 one can see the superior quality of the detection bounding boxes produced by the simulation trained network to those of the Cityscapes trained network. In particular, the simulation trained model produces less cluttered outputs. Code and data for reproducing the results is available at `https://github.com/umautobots/driving-in-the-matrix`

VI. Discussion

The results show that a simulation only training approach is viable for the purpose of classifying real world imagery. This is a powerful concept for self-driving car research, particularly the prospect of using a simulation environment to improve training and testing for increasingly larger and more widely deployed fleets of vehicles. The idea of using training data from simulation for deep learning has been



(a) Cityscapes



(b) Sim 200k

Fig. 4: Heatmaps of the training data's bounding box centroids. These plots show the frequency of cars in different locations in the image. Note the much larger spread of occurrence location in the simulated data (b) than in the real images of Cityscapes (a). In the proposed approach, cars are found in a wide area across the image aiding the network in capturing the diversity of real appearance.

explored in prior work, but it's impact has been mitigated due to the mixing of simulated and real images. In addition, we believe the model of using traditional training/test splits for modestly sized data sets may be leading to undesirable local minima. More explicitly, this pattern of training may be leading to overfitting or even memorization, which may hinder the development of generalizable models for object recognition.

The results achieved by the Cityscapes trained network, when evaluated on the KITTI data set, may in fact be due to overfitting since both data sets are from Germany and share similar cars and road structure. Though weather conditions varied marginally between the pair of data sets, they were both captured at the same time of day. Many open questions remain about the performance of deep learning networks when tested in much more diverse locations and weather conditions, as would be necessary for wide deployment.

Data set	Easy	Moderate	Hard
Sim 10k	0.5542	0.3828	0.2904
Sim 50k	**0.6856**	0.5008	0.3926
Sim 200k	0.6803	**0.5257**	**0.4207**
Cityscapes [20]	0.6247	0.4274	0.3566

TABLE II: This table presents the results of the mAP for 0.7 IoU on Easy, Moderate, and Hard Cars using a network never shown a KITTI image, but rather trained on Simulation and Cityscapes images respectively and then tested on all 7,481 KITTI images. Note our proposed approach, using only simulated car images, out performs real imagery (Cityscapes) on labels of all difficulties.

VII. Conclusions & Future Work

This paper presents a pipeline to gather data from a modern visual simulator with high realism to use as training data for object identification with deep learning networks. Networks trained using the simulated data were capable of achieving high levels of performance on real-world data without requiring any mixing with real-world training data set imagery. We highlight issues with data set bias in the way we train car detectors on modestly sized real data sets. Finally, we presented results that demonstrated improved performance from high numbers of simulated examples.

Future avenues for research include: confirming the performance of the simulated data across many different deep learning network architectures, deepening network architectures to maximize the impact of larger numbers of training examples, and using active learning or other approaches to maintain the high levels of performance with smaller subsets of simulation imagery. The goal of smaller data sets will both reduce training time and distill training image importance and help us to understand redundancy in this type of training data. Finally, we would like to explore greater complexity and specificity in the generation of simulation images.

References

[1] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and F. Li, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

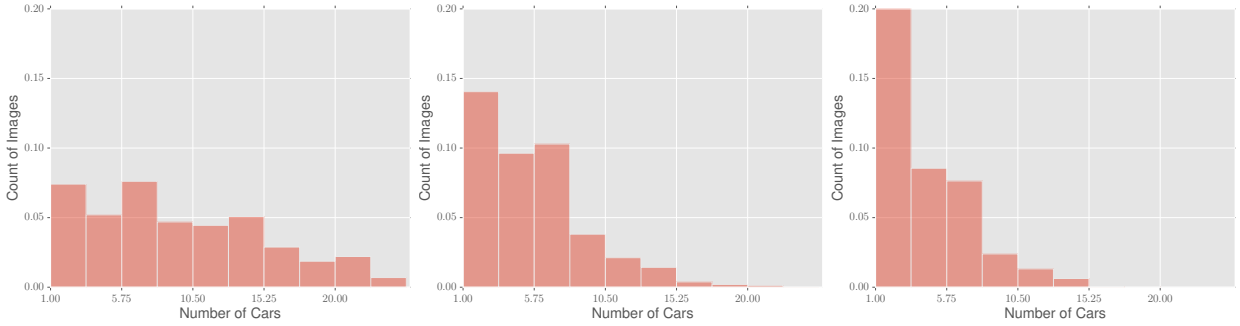
[2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[3] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman, "Labelme: A database and web-based tool for image annotation," *International Journal of Computer Vision*, vol. 77, no. 1-3, pp. 157–173, May 2008.

[4] L. von Ahn and L. Dabbish, "Labeling images with a computer game," in *Conference on Human Factors in Computing Systems (SIGCHI)*, ser. CHI '04, Vienna, Austria: ACM, 2004, pp. 319–326.

[5] L. von Ahn, R. Liu, and M. Blum, "Peekaboom: A game for locating objects in images," in *Conference on Human Factors in Computing Systems (SIGCHI)*, ser. CHI '06, New York, NY, USA: ACM, 2006, pp. 55–64.

[6] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.



(a) Cityscapes (b) Sim (c) KITTI

Fig. 5: These figures depict histograms of the number of detections per frame in the Cityscapes, Simulation, and KITTI data sets. Note the similarity of the simulation and KITTI data set distributions. This may aid the network trained using the simulation data set when evaluated on the KITTI data set.

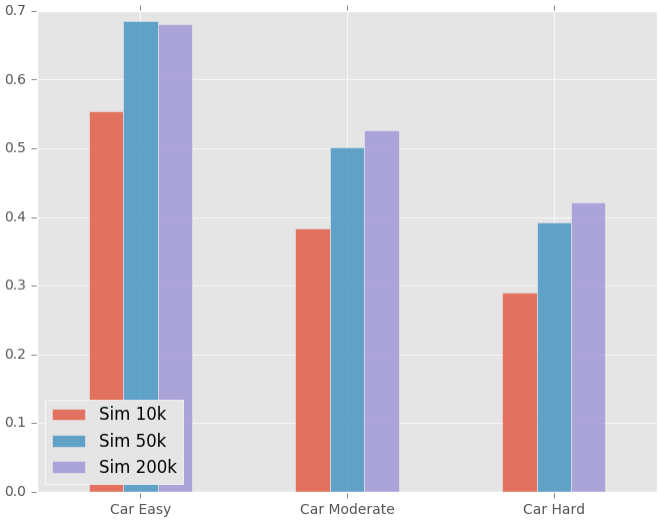


Fig. 6: Mean average precision for testing on the KITTI data set with results reported on the Easy, Moderate and Hard divisions of the labels. Note we use all 7,481 KITTI training images for validation. We do this as no KITTI imagery is used for training. We are focused on understanding how generalizable networks trained on other data is to the general problem of object classification.

(a) 10k

(b) 50k

(c) 200k

Fig. 7: A visualization of some representative detections from training on three increasing volumes of simulation data. Note how the performance improves in the larger training data sets. This is evaluated quantitatively in Table II, but the qualitative differences are illustrated in this figure.

[7] I. Lenz, R. Knepper, and A. Saxena, "Deepmpc: Learning deep latent features for model predictive control," in *Robotics Science and Systems (RSS)*, 2015.

[8] Y. LeCun, U. Muller, J. Ben, E. Cosatto, and B. Flepp, "Off-road obstacle avoidance through end-to-end learning," in *Advances in Neural Information Processing Systems (NIPS 2005)*, Y. Weiss, B. Scholkopf, and J. Platt, Ed., vol. 18, MIT Press, 2005.

[9] A. Torralba and A. A. Efros, "Unbiased look at dataset bias," in *CVPR'11*, Jun. 2011.

[10] L. Herranz, S. Jiang, and X. Li, "Scene recognition with cnns: Objects, scales and dataset bias," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016.

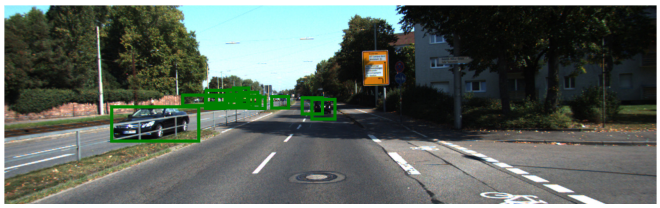
[11] A. Khosla, T. Zhou, T. Malisiewicz, A. A. Efros, and A. Torralba, "Undoing the damage of dataset bias," in *European Conference on Computer Vision*, Springer, 2012, pp. 158–171.

[12] J. Marin, D. Vázquez, D. Gerónimo, and A. M. López, "Learning appearance in virtual scenarios for pedestrian detection," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, IEEE, 2010, pp. 137–144.

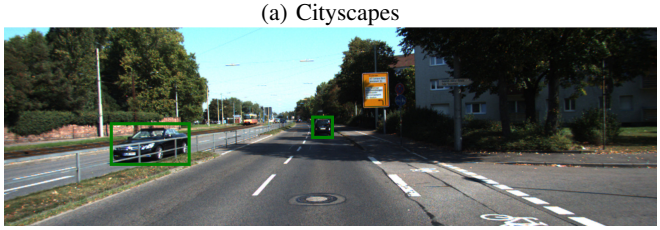
[13] J. Xu, D. Vázquez, A. M. López, J. Marín, and D. Ponsa, "Learning a part-based pedestrian detector in a virtual world," *IEEE Trans. Intelligent Transportation Systems*, vol. 15, no. 5, pp. 2121–2131, 2014.

[14] H. Hattori, V. Naresh Boddeti, K. M. Kitani, and T. Kanade, "Learning scene-specific pedestrian detectors without real data," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2015.

[15] H. Su, C. R. Qi, Y. Li, and L. J. Guibas, "Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views," in *The IEEE International Conference on Computer Vision (ICCV)*, Dec. 2015.



(a) Cityscapes



(b) 200k

Fig. 8: A visualization of some representative detections from training on Cityscapes and Sim 200k data sets. As seen in the figure, Sim 200k produces less cluttered outputs and better detections overall compared to Cityscapes.

[16] S. R. Richter, V. Vineet, S. Roth, and V. Koltun, "Playing for data: Ground truth from computer games," in *European Conference on Computer Vision (ECCV)*, 2016.

[17] G. J. Brostow, J. Shotton, J. Fauqueur, and R. Cipolla, "Segmentation and recognition using structure from motion point clouds," in *ECCV (1)*, 2008, pp. 44–57.

[18] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez, "The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016.

[19] A. S. Development, *Scripthook v*, 2016.

[20] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[21] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *CoRR*, vol. abs/1506.01497, 2015.

[22] T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, and Z. Zhang, "Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems," *CoRR*, vol. abs/1512.01274, 2015.