Crash To Not Crash: Playing Video Games To Predict Vehicle Collisions

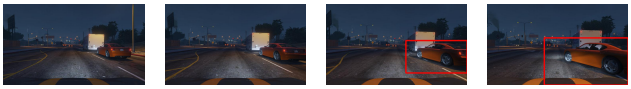
Kangwook Lee [*1] **Hoon Kim** [*1] **Changho Suh** [1]

Abstract

Today's vehicle collision prediction algorithms are rule-based, and have not benefited from the recent developments in deep learning. This is because it is almost impossible to collect a large amount of collision data from the real world. To address this challenge, we collect a large accident data set using a popular video game named GTA V. Using this accident data set, we develop efficient prediction algorithms based on modern CNN architectures. The performances of our algorithms are compared with the simple rule-based algorithms. We observe that the best CNN-based algorithm among several variants achieves the prediction accuracy of 96.2% while the best rule-based one achieves the accuracy of 89.6%. We also show that our approaches can identify the source of danger when a collision is predicted. Moreover, our approaches are shown to learn how the wheel angles, vehicle orientations, and distances affect the collision probability.

1. Introduction

Nearly 1.3 million people die in road crashes each year, and 3 out of 4 deaths are caused by incautious driving. Thus, one can save a significant number of lives by warning the driver ahead of a collision. The goal of collision prediction system (CPS) is to predict vehicle collisions as early as possible by using a variety of expensive sensors such as radar, LIDAR, and camera. While the recent success of deep learning has brought revolutions in numerous applications, most of the existing collision prediction algorithms still rely on rule-based ones. This is because deep learning algorithms require a large amount of training data but collecting sensor data from a large number of accident scenes has been considered impossible. In a recent work (Richter et al., 2016), it has been shown that video games or simulators are able to address the challenge of data scarcity since an arbitrarily huge amount of data can be collected from *the virtual world*. Not only that, they also demonstrate that their deep learning algorithms, trained with large synthetic data sets collected from the video games, can provide superior performances over those trained with small data sets that are collected from the real world.



(a) $t - 0.8\,\mathrm{s}$ (b) $t - 0.6\,\mathrm{s}$ (c) $t - 0.4\,\mathrm{s}$ (d) $t - 0.2\,\mathrm{s}$

Figure 1. We show 4 images taken *before* the accident which happens at time t. From left to right, the estimated collision probability for each image is $0.00, 0.45, 0.99$ and 1.00. Our algorithm successfully predicts the vehicle collision from these images. See (Lee et al., 2017) for similar demo videos.

In this work, we propose deep learning-based vehicle collision prediction algorithms, trained with a large data set collected from a popular video game named Grand Theft Auto V (GTA V). We first build our data generation framework, which we use to collect a large image data set. Our dataset, called `GTACrash`, consists of 26705 images taken from diverse accident scenes and 27325 images taken from nonaccident scenes. Using this data set, we train the collision prediction algorithms based on the state-of-the-art CNN (Convolutional Neural Network) architectures to predict whether or not the driver's car is going to collide within a certain period of time given an input image.

Our evaluation shows that the best CNN-based algorithm achieves the prediction accuracy of 96.2% while the best rule-based algorithm achieves the accuracy of 89.6%. We also show that the CNN-based algorithms can reliably identify the source of danger in accident images. Further, we show that the CNN-based algorithms learn how the wheel angles, vehicle orientations, and distances affect the collision probability.

[*]Equal contribution [1]KAIST, Daejeon, Korea. Correspondence to: Changho Suh <chsuh@kaist.ac.kr>.

Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, 2017. JMLR: W&CP.

In Fig. 1, we visualize the prediction results of our CNN-based algorithm for a sample accident scene. In this accident scene, the orange car on the next lane starts changing its lane at time $t - 0.8\,\text{s}$, and it eventually collides with the driver's car at time t. The 4 images shown in the figure are taken at times $t - 0.2\,\text{s}, t - 0.4\,\text{s}, \ldots, t - 0.8\,\text{s}$. For each image, our algorithm estimates the probability of collision as well as identifies the source of danger. Observe that our algorithm is able to predict the accident $t - 0.4\,\text{s}$ before the accident happens and identify the dangerous vehicle. See (Lee et al., 2017) for similar demo videos.

2. Related Work

Most of the existing collision prediction algorithms are based on various motion models such as physics-based ones, maneuver-based ones, and interaction-aware ones. See (Raut & Malik, 2014) for a complete survey. These approaches predict collisions by developing precise motion models, fitting the motions of the nearby vehicles, and estimating the future trajectories of them. These approaches, however, are applicable only when the past trajectories and maneuvers of nearby vehicles are available to the driver. This sets the fundamental limitation of those approaches: they require *all* the nearby vehicles to frequently exchange those information, making themselves dependent on the underlying communication infrastructure and hence vulnerable to communication errors. Different from these approaches, our algorithms do not assume any particular infrastructure. Instead, our algorithms predict collisions like human drivers: they perceive the driving scene and predict upcoming collisions.

Another line of related works is about self-driving and collision avoidance algorithms. In (Pomerleau, 1989), the authors demonstrate that a successfully trained neural network can mimic the way humans drive. Recently, deep learning has been applied to self-driving (Chen et al., 2015; Bojarski et al., 2016), achieving extraordinary performances. In (Bojarski et al., 2017), the authors explain how their deep neural networks perceive the scene, and show that they pay more attentions to key objects such as lane markings, nearby vehicles, etc. In (Kahn et al., 2017; Thorsson & Steinert, 2016), the authors apply reinforcement learning to develop collision avoidance systems (Kahn et al., 2017; Thorsson & Steinert, 2016). While these approaches have brought huge improvements in self-driving and collision avoidance algorithms, it is questionable whether they can handle unusual scenarios such as vehicle collisions, which are rarely observed in real data. Further, these approaches are still far from being stable enough to be deployed. Our approach is immediately deployable in today's vehicles since it is meant to *assist* human drivers, not to take over their roles.

A few recent works demonstrate that a large amount of realistic driving scenes can be collected from a virtual world via video games (Johnson-Roberson et al., 2016; Richter et al., 2016). They avoid the costly process of labeling images by collecting images from computer games, whose ground truth labels are readily obtainable. In this work, we go further by creating extreme environments such as vehicle accident scenes in the virtual world, which are hard to observe in the real world. To the best of our knowledge, our data set is the first driving data set that contains more than tens of thousands of accident images.

3. Data Collection

In this section, we describe our data generation framework and introduce our dataset, called `GTACrash`.

3.1. Data Generation

In order to collect a large amount of driving data, we first implement our own data generation framework based on Script Hook V (Blade, 2017), an open-source library that enables access to the low-level internal functions of GTA V. A few important functions available in our framework, which play key roles in our data generation process, are described as follows: `Start_Cruising(s)` lets the player's car cruise along the lane at speed `s`; `Get_Nearby_Vehicles()` returns the list of vehicles in the current scene of the game; `Set_Vehicle_Out_Of_Control(v)` makes the vehicle `v` drive out of control; `Is_My_Car_Colliding()` checks whether the driver's car is colliding; `Get_Bounding_Box(v)` returns the bounding box of the vehicle `v`; and `Set_Wheel_Angle(v, a)` sets the wheel angle of the specified vehicle `v` as `a`.

A data generation process begins with calling `Start_Cruising(s)`, making the player's car cruise along the lane at the specified speed. The player keeps exploring the virtual world until the end of the generation process. This is possible because the player's car has a full access to the internal states of the virtual world such as the road lanes and the map of the world. We now describe two different data generation modes: one for generating nonaccident scenes and the other for accident scenes. In both modes, the data generator collects screenshots of the game frames at the rate of 10 frames per second, and keeps only a subset of the collected screenshots according to certain rules.

Accident Scenes: We first illustrate how we collect

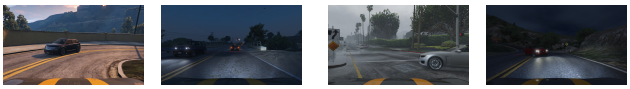


Figure 2. Sample images from our data set `GTACrash`. See (Lee et al., 2017) for more samples.

the accident data set. The data collector samples random times according to a Poisson process. At each of the sampled time, the collector makes one of the nearby vehicles start driving carelessly at random time, while the player keeps driving at the constant speed. We implement this by utilizing the functions available in our data generation framework as follows. We call `Get_Nearby_Vehicles()` to obtain the list of nearby vehicles at random time. We then choose one of those nearby vehicles at random, and make the chosen car start driving carelessly by calling `Set_Vehicle_Out_Of_Control(v)`. We observe that with high probability, the chosen car eventually ends up with crashing into the player's car. Note that the player continues driving at the constant speed merely following the lane, and it does not attempt to avoid the collision. Whether or not a car accident actually happens can be checked by using `Is_My_Car_Colliding()`. When an accident is detected in frame i, we take the 5 screenshots of the 5 consecutive frames $i-1, i-2, \ldots, i-5$. That is, for each accident scene, we collect 5 images that are taken $i \times 0.1\,\mathrm{s}$ *before* the collision happens for $i = 1, 2, \ldots, 5$. For each image, we also collect the ground-truth bounding boxes of the vehicles via `Get_Bounding_Box(v)`. Further, we also store the id of the vehicle that is out of control in each image.

Nonaccident Scenes: The nonaccident scenes are collected in a similar way. The only difference is that the other vehicles in the virtual world drive normally at all time. For each sampled driving scene, the screenshots of the 5 consecutive frames are collected. Further, we annotate each image with the ground-truth bounding boxes of visible objects and the number of visible vehicles in the image. We define $\mathcal{G}_K^0$ as the set of nonaccident images with K visible vehicles for $K \in \{0, 1, 2, \ldots\}$. In our experiments, we observe that $|\mathcal{G}_0^0|$ is much larger compared to $|\mathcal{G}_K^0|$ for $K > 1$. In order to increase the diversity of the data set, we discard 50% of the images in $\mathcal{G}_0^0$ at random.

3.2. Overview of `GTACrash`

Our dataset `GTACrash` consists of 5465 nonaccident scenes (5465×5 nonaccident frames) and 5341 accident scenes (5341×5 accident frames). We randomly sample one fifth of the data set as the test set. Note that it takes only about 48 hours on a single computer running GTA V to collect the dataset.

The driving scenes are randomly sampled from about 48 hours long game playing, which is equivalent to about 60 days long driving in the virtual world. This makes the driving scenes of our data set extremely diverse: Those images are taken in arterial roads and highways, in days and nights, and on sunny, rainy, and snowy days. Fig. 2 shows some sample driving scenes from `GTACrash`. The first four figures are nonaccident images, and the others are accident images. For more samples, see (Lee et al., 2017).

4. Collision Prediction Algorithms

Here, we propose collision prediction algorithms based on the state-of-the-art CNN architectures. We also introduce simple rule-based algorithms, which are based on object detection algorithms.

4.1. CNN-based Algorithms

We present a class of CNN-based prediction algorithms, whose network architectures are based on those of AlexNet (Krizhevsky et al., 2012), VGG16 (Simonyan & Zisserman, 2014), and ResNet50 (He et al., 2016). More specifically, we modify the networks by replacing the FC (fully-connected) layers of these CNN architectures[1] with two FC layers. The first hidden layer is of size m, and the last layer of size 2 corresponds to the binary classification results. For each of the CNN architecture, we test the following training strategies. The first approach is basic: it initializes all the parameters in a network with random values, and trains the network from scratch. If our data set is large enough to train the large network from scratch without overfitting, this has potential to achieve the best performance. Another approach is the *transfer learning (tl)* approach (Pan & Yang, 2010), which will be useful if our data set is not large enough to train the network from scratch but sufficient for fine-tuning the weights. The parameters of the convolution layers are initialized with those trained with ImageNet data set, while the parameters of the FC layers are randomly initialized. Then, the entire network is trained with the new data set. The last approach is the *deep feature (df)* approach (Donahue et al., 2014; Sharif Razavian et al., 2014): by viewing the output of the convolution part as a generic image feature, we train the FC layers only while keeping the parameters of the convolution part as those trained with ImageNet. Note that the deep feature approach can be viewed as a special case of the transfer learning approach: here we call it the df approach for short.

[1] The layers after pool5 for AlexNet and VGG16 and the last FC layer of ResNet50.

Figure 3. Three shapes of the danger zone. From left: triangle, polygon with 7 sides, and trapezoid.

4.2. Detection-based Algorithms

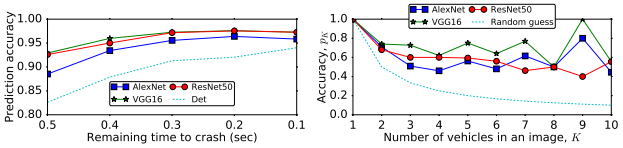
As a rule-based approach to collision prediction, we present simple detection-based algorithms. The detection-based algorithms first detect nearby objects, and decide whether the nearby objects are within a proximity of the driver's car. If the intersection area between the bounding box of a nearby object and a certain area around the driver, which we call *the danger zone*, is larger than a certain threshold, one can classify the scene as an accident scene. Note that its performance is highly affected by the performance of the underlying object detection algorithm. In this work, we consider an oracle algorithm: that is, we provide the algorithm with the ground-truth bounding boxes for all nearby objects for all scenes. It is clear that the performance of this oracle algorithm sets an upper bound on the performance of the detection-based algorithms equipped with any detection algorithms. The most important factors that affect the prediction performance is the shape and the size of the danger zone. We consider three shapes for the danger zone: triangle, polygon with 7 sides, and trapezoid. We show the shapes of the danger zones in Fig. 3. (The size of each danger zone is optimized to maximize the prediction performance.) We call the detection-based algorithm 'Detection(i)' if the ith shape in the figure is used for the danger zone.

5. Experimental Results

In this section, we evaluate the performances of the proposed algorithms. Throughout this section, the size of hidden layer is fixed as $m = 100$. We train the CNN-based algorithms using AdamOptimizer with mini-batches of size 64. For the learning rates, we use 10^{-3} for the FC layers and 10^{-4} for the inner layers. We first show that the CNN-based algorithms significantly outperform the simple detection-based algorithms in terms of prediction performance. More interestingly, we show that the CNN-based algorithms are able to learn a few important clues to a collision: the wheel angles of the other vehicles, the distances to them, and the orientations of them.

5.1. Prediction Performance

The prediction performances of the algorithms are shown in Table. 1. The first row compares the classification accuracies of the algorithms. One can observe that VGG16 with transfer learning (tl) achieves the best accuracy of 0.962, which is a huge improvement over the accuracy of 0.896, achieved by the best detection algorithm. The second row reports the area under curve (AUC) of the receiver operating characteristic (ROC) curve. We observe that the VGG16 with the basic strategy achieves the best AUC of 0.986 while the best detection-based one achieves the AUC of 0.922.



(a) Acc. vs time to crash (b) Identification acc. vs K

Figure 4. Prediction and identification accuracies.

Figure 5. Source identification examples.

Recall that each accident scene consists of 5 consecutive frames, where the first frame is taken 0.5 s before the accident happens and the last frame is taken 0.1 s before the accident. Clearly, the collision prediction task is easier with the images of the frames that are closer to the accident. Fig. 4a shows how the classification accuracy changes as a function of 'the remaining time to crash'. Observe that the performance gap between the CNN-based algorithms and the detection-based algorithms increases as the remaining time to crash increases, implying that the CNN-based algorithms can predict a collision earlier than the others.

5.2. Identification of Dangerous Vehicles

In addition to raising the alarm, a CPS system may also pinpoint the source of the danger. For instance, it can highlight the source using the head-up display, i.e., a transparent display that can render computer graphics without blocking the frontal view.

The detection-based algorithms can directly locate the source of the danger: the car that intersects the danger zone most is the source of the danger. On the other hand, the CNN-based algorithms cannot immediately locate the source of the danger. In order to enhance the interpretability of the CNN-based algorithms, we propose the following two-stage algorithm, inspired by the occlusion approach proposed in (Zeiler & Fergus, 2014). The first stage of the algorithm is the object detection stage: it first detects all the objects in the front view such as vehicles, pedestrians, animals, etc. Assume that $K \geq 1$ objects are detected in the image. It then masks out each of the K objects by patching the bounding box of the object with the average pixel values. Then, each of the patched images is fed to

Table 1. Prediction Performance. For the CNN-based algorithms, 'b' denotes the basic learning strategy, 'tl' denotes the transfer learning approach, and 'df' denotes the deep feature approach.

	Detection			AlexNet			VGG16			ResNet50		
Metric	1	2	3	b	tl	df	b	tl	df	b	tl	df
Accuracy	0.875	0.896	0.881	0.912	0.940	0.897	0.952	**0.962**	0.923	0.905	0.960	0.870
AUC	0.907	0.922	0.904	0.964	0.977	0.956	**0.986**	0.966	0.984	0.956	**0.986**	0.939

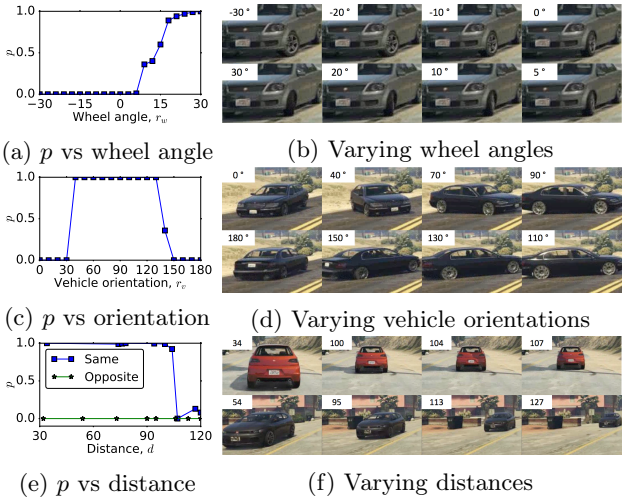
the CNN-based algorithm. Denote by p_i the output for the ith image for $1 \leq i \leq K$. Similarly, denote by p_0 the output for the original image. The two-stage algorithm then finds the object whose presence increases the collision probability most among the K objects. That is, if $\hat{x} = \arg\max_{1 \leq i \leq K}(p_0 - p_i)$, the $\hat{x}$th object is chosen as the source of danger.

We now evaluate the source identification performances of the detection-based algorithms and the proposed two-stage algorithm. Recall that each image is annotated with the ground truth source of danger, which is the car that is driving out of control in the image. We measure the identification accuracy using these labels as follows. We define $\mathcal{G}_K^1$ as the set of accident images with K visible vehicles for $K \in \{0, 1, 2, \ldots\}$. Further, denote the source of danger in image z by x_z, and the output of an identification algorithm by $\hat{x}_z$. For fixed K, the empirical identification accuracy is defined as $q_K = \sum_{z \in \mathcal{G}_K^1} \mathbf{1}(\hat{x}_z = x_z)/|\mathcal{G}_K^1|$. Clearly, a random guess can achieve $q_K = 1/K$.

Shown in Fig. 5 are some examples of the source identification results. If $p_0 - p_i < 0.01$, the ith vehicle is annotated with a green bounding box. Otherwise, we draw a red bounding box, annotated with the value of $p_o - p_i$: the red bounding boxes denote dangerous objects. For demo videos, see (Lee et al., 2017).

We also test how well the CNN-based algorithms identify the source of danger. Note that when there are no vehicles in the danger zone, the detection-based algorithms need to guess the source of danger at random while our two-stage algorithms still have a chance to correctly identify the source. In order to see this, we evaluate the source identification accuracies of our two-stage algorithms[2] on the accident images where no vehicles exist in the danger zone. Shown in Fig. 4b are the results. We observe that the identification accuracies of the two-stage algorithms decay much slower than those of the other algorithms.

[2]For the two-stage algorithm, we provide the second stage of the algorithm with the ground-truth bounding boxes. Note that in practice, one can run an efficient object detection algorithm to obtain the bounding boxes.



(a) p vs wheel angle (b) Varying wheel angles

(c) p vs orientation (d) Varying vehicle orientations

(e) p vs distance (f) Varying distances

Figure 6. The rules learned by the CNN-based algorithms.

5.3. The Rules Learned by the CNN-based Algorithms

In the previous sections, we observed that the CNN-based algorithms constantly outperform the simple detection-based algorithms. A natural question then arises: What are the implicit prediction rules that are learned by the CNN-based algorithms? For instance, one may wonder whether the CNN-based prediction algorithms are able to detect subtle evidences such as abnormal wheel angles of other vehicles, unusual orientation of them, etc. In this section, we show that the CNN-based algorithms learn to output a higher collision probability if

- (wheel angle) the wheel angles of the other cars are not aligned with their lanes, or
- (vehicle orientation) the other cars are facing toward the driver, or
- (distance) the nearby vehicles are closely located.

Wheel angle: We first collect a set of random driving images. For each driving scene, we choose one of the nearby vehicles on the other side of the lanes. We then generate a sequence of driving images by varying the wheel angle of the chosen car. This can be done by using the function `set_wheel_angle()` of our data generation framework. We then measure the output of

the CNN-based algorithms and observe how the wheel angles are correlated with the prediction output. The wheel angle r_w is measured with respect to the direction of the lane: $r_w = 0°$ if the wheel is aligned with the lane; $r_w = +30°$ if the wheel is rotated by 30° toward the driver's lane; and $r_w = -30°$ if the wheel is rotated by 30° toward the sidewalk.

Shown in Fig. 6a are the average prediction probabilities as a function of the wheel angle. We clearly observe the positive correlation between the wheel angle and the predicted probabilities of the CNN-based algorithms. Note that the outputs of the detection-based algorithms do not alter at all when the wheel angle changes. Some sample images are shown in Fig. 6b.

Vehicle orientation: We also measure the correlation between the vehicle orientation and the prediction probability. The vehicle orientation r_v is measured as the angle between the direction of the lane and the orientation of the vehicle: $r_v = 0°$ if the car is facing the direction of the lane, and $r_v = 90°$ if the car is facing the driver's lane. The average prediction probability as a function of the vehicle orientation is plotted in Fig. 6c. We observe that the predicted probability is close to one when $40° \leq r_v \leq 130°$: this makes sense because it is probable that the vehicle crosses over the central line, and collides with the driver's car.

Distance: Another interesting observation is the relationship between the distance to the nearby vehicle and the predicted probability. Shown in Fig. 6e is the predicted probability as a function of the distance to the nearby car, measured in pixels. Observe that when the nearby vehicle is on the same lane ahead of the driver's car, the predicted probability decreases as the other vehicle moves farther. On the other hand, when the nearby vehicle is on the opposite lane, the predicted probability is always zero, regardless of the distance to the vehicle. This makes a perfect sense since the distances to the cars driving normally on the opposite lane should not affect the collision probability.

5.4. Prediction Performance on Real Images

Figure 7. Sample images in the data set of real images.

While the CNN-based algorithms show superior performances on the driving images collected from the virtual world, it remains questionable whether these algorithms can predict accidents from the real images. To evaluate the generalization performance of CNN-based algorithms, we collect a data set of real images as follows. We first collect 102 video clips on car accident from YouTube. For each clip, we collect two accident images and two nonaccident images: Denoting the time at which the accident happens by t, we collect the frames at time $t - 0.2\,\text{s}$ and $t - 0.4\,\text{s}$ as accident images and the frames at time 0 and $0.2\,\text{s}$ as nonaccident ones. See Fig. 7 for some sample images.

The prediction performance of the VGG16(tl) algorithm is evaluated on these real images: The prediction accuracy is measured as 0.71, and the AUC is measured as 0.63. The poor performance seems to be due to the low quality of the data set. We believe that transfer learning can significantly improve the performance on real images: See Sec. 6 for more discussion.

6. Conclusion and Discussion

In this work, we collect a large image data set using video games, and propose a CNN-based algorithm for collision prediction. Our approach is shown to 1) predict collisions ahead of time, 2) detect the source of danger, and 3) learn a few interesting rules. We conclude the paper by presenting future directions.

Richer data sets: Video games are a source of infinite data, and our `GTACrash` is only tip of the iceberg. A richer data set can be generated by collecting rear view images, views from multiple cars, simulated radar and LIDAR sensor outputs, velocities of the nearby objects, etc. Further, one can also vary the player's driving policy and the other vehicles' driving policies. The diversity of the data set can be further augmented by adding more vehicle models and new virtual cities.

Alternative approaches: While we use a single image for collision prediction, one may design prediction algorithms which take a sequence of consecutive images as input. One may estimate optical flows from consecutive images and train a CNN with the optical flows, possibly via deep learning (Fischer et al., 2015). Another viable option is to use recurrent neural network (RNN) (Hochreiter & Schmidhuber, 1997) that takes multiple frames. Further, one may generate future images via a generative adversarial network (GAN) and then predict collisions with generated future images.

Knowledge transfer to the real world: In (Richter et al., 2016), the authors show that the semantic segmentation model trained with 1/3 of the real data set and a large virtual data set performs as good as the model trained with the entire real data set: The amount of real data can be reduced by transferring the knowledge learned from the virtual world. We plan to study whether the knowledge we collect in this work can be transferred well to the real world.

References

Blade, Alexander. Script hook v. http://www.dev-c.com/gtav/scripthookv/, 2017. Accessed: 2017-06-06.

Bojarski, Mariusz, Del Testa, Davide, Dworakowski, Daniel, Firner, Bernhard, Flepp, Beat, Goyal, Prasoon, Jackel, Lawrence D, Monfort, Mathew, Muller, Urs, Zhang, Jiakai, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

Bojarski, Mariusz, Yeres, Philip, Choromanska, Anna, Choromanski, Krzysztof, Firner, Bernhard, Jackel, Lawrence, and Muller, Urs. Explaining how a deep neural network trained with end-to-end learning steers a car. *arXiv preprint arXiv:1704.07911*, 2017.

Chen, Chenyi, Seff, Ari, Kornhauser, Alain, and Xiao, Jianxiong. Deepdriving: Learning affordance for direct perception in autonomous driving. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2722–2730, 2015.

Donahue, Jeff, Jia, Yangqing, Vinyals, Oriol, Hoffman, Judy, Zhang, Ning, Tzeng, Eric, and Darrell, Trevor. Decaf: A deep convolutional activation feature for generic visual recognition. In *Icml*, volume 32, pp. 647–655, 2014.

Fischer, Philipp, Dosovitskiy, Alexey, Ilg, Eddy, Häusser, Philip, Hazırbaş, Caner, Golkov, Vladimir, van der Smagt, Patrick, Cremers, Daniel, and Brox, Thomas. Flownet: Learning optical flow with convolutional networks. *arXiv preprint arXiv:1504.06852*, 2015.

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

Hochreiter, Sepp and Schmidhuber, Jürgen. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Johnson-Roberson, Matthew, Barto, Charles, Mehta, Rounak, Sridhar, Sharath Nittur, and Vasudevan, Ram. Driving in the matrix: Can virtual worlds replace human-generated annotations for real world tasks? *arXiv preprint arXiv:1610.01983*, 2016.

Kahn, Gregory, Villaflor, Adam, Pong, Vitchyr, Abbeel, Pieter, and Levine, Sergey. Uncertainty-aware reinforcement learning for collision avoidance. *arXiv preprint arXiv:1702.01182*, 2017.

Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Lee, Kangwook, Kim, Hoon, and Suh, Changho. Online supplementary material for the submission. https://sites.google.com/view/gtacrashanon, 2017. Accessed: 2017-06-06.

Pan, S. J. and Yang, Q. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, Oct 2010. ISSN 1041-4347. doi: 10.1109/TKDE.2009.191.

Pomerleau, Dean A. Advances in neural information processing systems 1. chapter ALVINN: An Autonomous Land Vehicle in a Neural Network, pp. 305–313. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1989. ISBN 1-558-60015-9. URL http://dl.acm.org/citation.cfm?id=89851.89891.

Raut, Swati B and Malik, LG. Survey on vehicle collision prediction in vanet. In *Computational Intelligence and Computing Research (ICCIC), 2014 IEEE International Conference on*, pp. 1–5. IEEE, 2014.

Richter, Stephan R, Vineet, Vibhav, Roth, Stefan, and Koltun, Vladlen. Playing for data: Ground truth from computer games. In *European Conference on Computer Vision*, pp. 102–118. Springer, 2016.

Sharif Razavian, Ali, Azizpour, Hossein, Sullivan, Josephine, and Carlsson, Stefan. Cnn features off-the-shelf: An astounding baseline for recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2014.

Simonyan, Karen and Zisserman, Andrew. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Thorsson, Jonathan Leiditz and Steinert, Olof. Neural networks for collision avoidance. 2016.

Zeiler, Matthew D and Fergus, Rob. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pp. 818–833. Springer, 2014.