# Bridging the Reality Gap: Investigation of Deep Convolution Neural Networks Ability to Learn from a Combination of Real and Synthetic Data

Omar Gamal, Keshavraj Rameshbabu, Mohamed Imran and Hubert Roth
*Institute of Automatic Control Engineering, University of Siegen, Hölderlinstraße 3, Siegen, Germany*

Keywords: Reality Gap, Domain Transfer, Dataset Scarcity, Artificial Data, Convolution Neural Networks.

Abstract: Recent advances in data-driven approaches especially deep learning and its application on visual imagery have drawn a lot of attention in recent years. The lack of training data, however, highly affects the model accuracy and its ability to generalize to unseen scenarios. Simulators are emerging as a promising alternative source of data, especially for vision-based applications. Nevertheless, they still lack the visual and physical properties of the real world. Recent works have shown promising approaches to close the reality gap and transfer the knowledge obtained in simulation to the real world. This paper investigates Convolution Neural Networks (CNNs) ability to generalize and learn from a mixture of real and synthetic data to overcome dataset scarcity and domain transfer problems. The evaluation results indicate that the CNN models trained with real and simulation data generalize to both simulation and real environments. However, models trained with only real or simulation data fails drastically when it is transferred to an unseen target environment. Furthermore, the utilization of simulation data has improved model accuracy significantly.



## 1 INTRODUCTION

Deep Convolution Neural Networks (CNNs) uses multiple layers to automatically learn the hierarchical representation from the raw input data. This process, however, requires a massive amount of training data which in most cases is difficult to find, e.g. in medical and education domains. The amount of data required during training depends mainly on the network complexity, type of data, existing gaps between data samples, and whether we are training from scratch or fine-tuning a pre-trained model (Tkacz, 2005).

The CNN models trained with a low amount of data are very poor in terms of performance and cannot generalize to unseen scenarios (Yu and Yali, 2011). Fine-tuning of pre-trained models and usage of synthetic data, i.e. data collected from simulators are promising solutions for dataset scarcity. Simulators can provide the domain with an infinite amount of data samples, however, they do not have a perfect representation of the real world in both visual and physical properties (Bousmalis and Levine, 2017). This creates a huge gap between both environments which is often called the "reality gap". The reality gap has been addressed by researchers in different works and there exist several strategies for bridging the gap between simulation and reality. One of the strategies adopted is to develop high-quality simulators that represent the real-world visual and physical properties as close as possible. Furthermore, rendering real-world images has shown significant improvement in performance (Peng, Sun, Ali, and Saenko, 2014; Stark, Goesele, and Schiele, 2010; Su, Qi, Li, and Guibas, 2015).

Domain randomization is one of the recent and promising techniques used to bridge the gap between simulation and reality. Sadeghi and Levine (2016) showed that training deep reinforcement learning algorithms only on simulation data can generalize to the real world by randomizing the simulation environment. Similarly, Tobin et al. (2017) showed that randomizing the simulation environment using non-realistic textures can generalize a deep neural network model to the real environment.

Retraining the CNN model from scratch or fine-tuning a pre-trained model in the unseen target environment has shown also a significant increase in performance. For example, Tzeng, Hoffman, Zhang, Saenko, and Darrell (2014) proposed a new CNN model architecture with adaption layer and domain



Gamal, O., Rameshbabu, K., Imran, M. and Roth, H.
Bridging the Reality Gap: Investigation of Deep Convolution Neural Networks Ability to Learn from a Combination of Real and Synthetic Data.
DOI: 10.5220/0009830804460454
In *Proceedings of the 17th International Conference on Informatics in Control, Automation and Robotics (ICINCO 2020)*, pages 446-454
ISBN: 978-989-758-442-8

confusion loss to automatically learn meaningful and domain invariant representations. Li, Wang, Shi, Liu, and Hou (2016) presented an adaptive batch normalization approach that ease model transfer to an unseen target domain. A variety of other studies have shown a significant increase in CNN model performance when trained in the unseen target environment (see, e.g., Duan, Xu, and Tsang, 2012; Hoffman, et al., 2014; Kulis, Saenko, and Darrell, 2011; Yosinski, Clune, Bengio, and Lipson, 2014).

In robotics field, however, domain transfer is rather difficult to apply, especially when vision sensors are used to perceive the surrounding environment. Only a few studies have examined domain transfer in robotics and showed successful results (see, e.g., Cutler, Walsh, and How, 2014; James, Davison, and Johns, 2017; Loquercio, et al., 2019; Tobin, et al., 2017; Yan, Frosio, Tyree, and Kautz, 2017; Zhang, Leitner, Milford, and Corke, 2016). Therefore, there exist fewer approaches to transfer the knowledge obtained in simulation to reality (Sünderhauf et al., 2018).

To our knowledge, most of the research in the robotics field employs either simulation or real data but not a mixture of both. This paper investigates Convolution Neural Networks (CNNs) ability to learn from a mixture of real and artificial data when trained from scratch and fine-tuned to fit the dataset classes of the respective task. Each CNN model is trained six times wherein each time a different combination of the collected real and simulation datasets is used. The learning ability of the trained CNN models is evaluated using classification models metrics and deployment of models frozen inference graphs in both environments, i.e. real and simulation environments.

The work of Bayraktar, Yigit, and Boyraz (2019) is the most similar to our idea, however, their focus is mainly on object detection. The authors proposed their own dataset ‘‘ADORESet’’ which includes 30 classes with 2500 real plus 750 artificial images per class. In their experiments, they used the dataset to fine-tune four different pre-trained models, however, they did not consider training models from scratch. Furthermore, they did not deploy the trained models to real and simulation environments.

## 2 METHODOLOGY

In this work, we will investigate the ability of CNN models to generalize and learn from a mixture of real and artificial data to overcome dataset scarcity and domain transfer problems. We will examine the learning ability of CNN models when trained from scratch and fine-tuned to fit the dataset classes of lane following task. In lane following, the models are trained to infer the steering angle and velocity required to drive a Radio-controlled (RC) car model kit autonomously on the track.

For dataset collection and inference stages, we created physical and simulation environments for the lane following task. The simulation environment is created using the robot simulator CoppeliaSim and made to be as close as possible to the physical environment. To ease the dataset collection in both environments we used computer vision to define the centreline coordinates of the track and generate a path from start to endpoint. We then utilized a geometric path tracking system developed in an earlier project (Gamal, Imran, Roth, and Wahrburg, 2020) to follow the generated path. To localize the vehicle in the environment the Apriltags ROS wrapper is used. Figure 1 illustrates the geometric path tracking system used for lane tracking task.

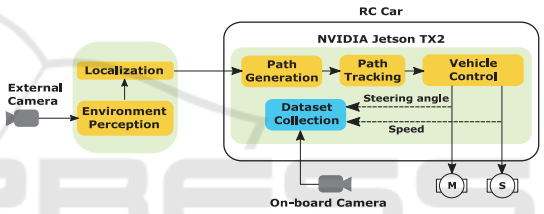


Figure 1: Geometric path tracking system.

To evaluate the learning ability of the CNN models six datasets are created wherein four of them 33% of the dataset collected from the real-world environment is kept fixed based on the assumption that we have a very limited dataset. In these four datasets, the number of incorporated simulation data samples is varied. More specifically, each CNN model will be trained for six times wherein each time the dataset is changed as follows; 0% real word data + 100% of simulation data, 100% real-world data + 0% simulation data, 33% real-world data + 0% simulation data, 33% real-world data + 33% simulation data, 33% real-world data + 66% simulation data, and finally 33% real-world data + 100% simulation data. To measure the models' performance and their learning ability classification models evaluation metrics are used. Furthermore, the frozen inference graphs of the models trained with all datasets combinations are deployed on both environments, i.e. simulation and real-world environments.

# 3 SYSTEM SETUP

In this section, we will discuss the procedure followed to setup the physical and simulation environments for lane tracking task. Furthermore, we will introduce the platform used in the real-world environment for dataset collection and CNN models' inference.

## 3.1 Test Platform

The test platform is an RC car model kit with Ackerman steering mechanism, see Figure 2-b.

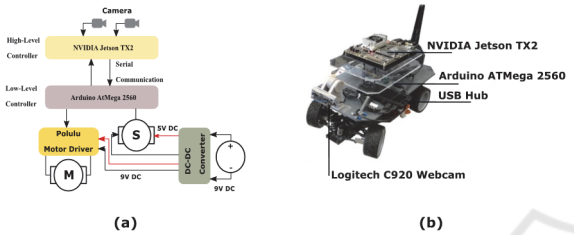


Figure 2: (a) Vehicle control system (b) Test platform.

The platform features a DC motor belt drive system and a servo motor for controlling the steering angle. The servo motor angle ranges from 0° to 180 ° however it is restricted with Ackerman steering mechanism range. A 90° steering angle steers the front wheels to face forward direction, an angle below 90° steers it to the right and above 90° to the left. Figure 2-a illustrates the vehicle control system where the low-level controller (Arduino Mega 2560 board) is used for sensor data collection, motors control, and bidirectional communication with the high-level controller. The high-level control is the robot's onboard computer (NVIDIA Jetson TX2 developer kit). It is used for high processing tasks (e.g. CNN models' deployment) and to provide an interface with sensors and hardware components, whose data transmissions standards are found difficult to be handled by the Arduino microcontroller. Furthermore, the car is equipped with a Logitech C920 webcam for dataset collection and CNN models' inference. The camera is placed facing the heading direction of the vehicle and tilted with an angle of 70° to keep the focus on the region of interest, i.e. track lanes.

## 3.2 Test Environments

As stated earlier two test environments are setup for dataset collection and CNN models' inference. These are real and simulation environments. In the next subsections, we will discuss the procedures followed to setup both environments.

### 3.2.1 Real Environment Setup

To prepare the lane track, we used black carton sheets and white tape to mark the track lanes. Figure 3 depicts physical environments setup.

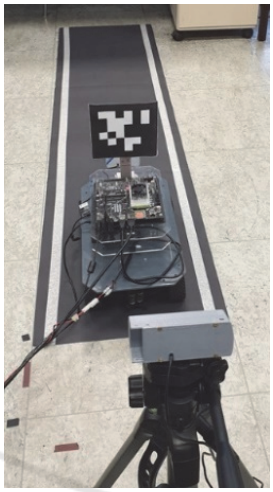

Figure 3: Lane tracking environment.

The ROS wrapper of the AprilTag 3 system is used for localizing the vehicle in the environment using external Logitech C920 webcam. The pose of all April tags present in the field of view of the camera is tracked by the tf ROS package which allows us to get the transform between any two coordinate frames in the system transformation tree. The position and orientation of the detected tag are fed to the geometric path tracking algorithm to infer the required vehicle speed and steering angle to drive the vehicle autonomously on the generated path.

### 3.2.2 Simulation Environment Setup

The robot simulator CoppeliaSim formally known as V-REP is used to create the lane tracking simulation environment. The car model used in the robot simulator is a simple Ackermann Steering model provided by CoppeliaSim, which is similar to the test platform used in this work. A vision camera is placed on top of the model for dataset collection and CNN models' inference.

To add a new environment in CoppeliaSim, a Unified Robot Description Format (URDF) model of the environment is required. We used Solidworks to model the environment and sw_urdf_exporter add-in to convert the environment model into URDF model

format. Figures 4 illustrates the lane tracking simulation environment.

Figure 4: Lane tracking simulation environment.

The CoppeliaSim platform provides dummy objects to track the position and orientation of the robot within the simulation environment. In the lane following simulation environment, we used three dummy objects to define the vehicle position, start, and endpoints on the track. To use the same lane tracking algorithms in the simulation environment we used CoppeliaSim remote API to establish a communication channel between CoppeliaSim and our External client application, i.e. Python scripts.

# 4 CNN MODELS DESIGN AND TRAINING

To investigate the learning ability of Convolution Neural Networks two models are evaluated where the first is trained from scratch and the second is fine-tuned to fit the dataset classes of the lane tracking task. To train these models two datasets are collected; namely simulation and real-world datasets.

In the next subsections, we will discuss the dataset collection and CNN model architecture design and training.

## 4.1 CNN Model Architecture

The chosen CNN model architecture was designed in one of our earlier projects related to lane tracking which makes it an ideal candidate for our study. The network architecture was inspired by different CNN model architectures such as the Inception model by Szegedy et al. (2015) and Network In Network structure by Lin, Chen, and Yan (2013). Figure 5 illustrates the pre-trained model architecture. The network model has 126,058 trainable parameters and takes an input image of size (320 X 240 X 3). It consists of six convolution layers, four Maxpooling layers, one dropout layer, flatten layer, and two fully connected layers. The first convolution layer consists of 64 kernels of size 11X11 with stride 4X4 and followed by a Maxpooling layer with a pool size of 2 X 2. The second convolution layer has 128 kernels of size 7X7 with stride 2X2 and followed by a Maxpooling layer with a pool size of 2 X 2. The Max-pooling layer is then followed by two inception modules with different kernel sizes. These are 5x5, 3x3, and 3X3, 1X1 respectively. Following this, a Maxpooling layer and one additional inception module with a kernel size of 1x1 combined with average pooling. Finally, we have a network in network and two fully connected layers.

## 4.2 Dataset Collection and Preparation

In lane following, the steering angle range used is 80° to 100° and divided into 3 classes. The chosen range will ensure that the vehicle doesn't deviate from the center of the track because of the small field of view of the camera. Owing to the large weight that the RC Car carries, the speed of the vehicle is kept fixed at 100% throttle value ''full speed'' so that the DC motor is able to drive the vehicle. Table 1 illustrates the dataset class labels of the lane following task where F means forward and S stop.

Table 1: Lane tracking dataset class labels.

| No. | Class | Turning Angle | Velocity |
|---|---|---|---|
| 0 | 80F100 | 80 | 100% |
| 1 | 90F100 | 90 | 100% |
| 2 | 100F100 | 100 | 100% |
| 3 | 090S000 | 0 | 100% |

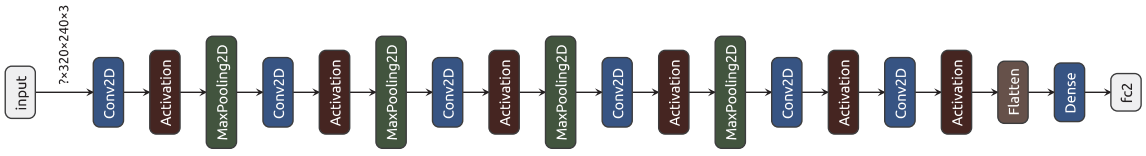


Figure 5: Lane tracking CNN model architecture.

In dataset collection, the geometric path tracking system is used to drive the car autonomously in both simulation and real environments and simultaneously collects, labels, and store the data. Two datasets are created, i.e. real and simulation datasets where each contains around 30,000 labeled images. Figures 6 illustrates the number of images collected per class in both environments.

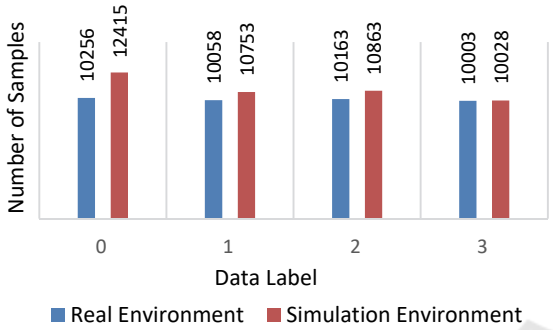


Figure 6: Dataset collected for the lane tracking task.

## 4.3 CNN Models Training and Fine Tuning

As mentioned earlier the two CNN models will be trained for six times wherein each the dataset is changed as follows:

a) 0% real word data + 100% simulation data

b) 100% real-world data + 0% simulation data

c) 33% real-world data + 0% simulation data

d) 33% real-world data + 33% simulation data

e) 33% real-world data + 66% simulation data

f) 33% real-world data + 100% simulation data

The datasets are further divided into training and validation datasets. The training and validation datasets percentage are chosen to be 90% and 10% of the original dataset respectively. The CNN model used in this work is a pre-trained model with ten classes. The model will be fine-tuned to fit the four classes of the lane tracking task. To fine-tune the model, we truncated the last fully connected layer of the pre-trained network and replaced it with our own fully connected layer with four units and softmax activation. To train the same CNN model from scratch the complete model parameters are randomly initialized and trained to find the optimal parameter set that maps the inputs to their associated targets.

The networks are trained using Adam optimizer and binary cross-entropy loss function. The batch size used for training is 32 and the learning rate is 0.001. During the training phase, we used model checkpoints to continually save the model along with the corresponding epoch number. Figures 7, 8 depict the training loss and validation graphs of the trained

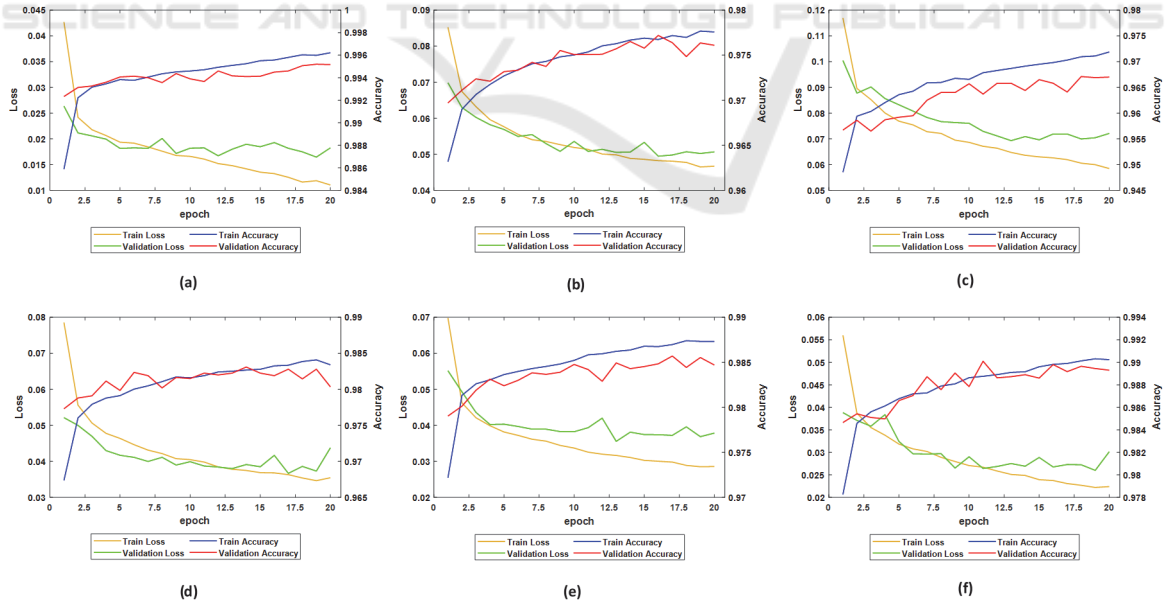


Figure 7: Fine-tuned CNN models training graphs (a) 100% of Simulation dataset (b) 100% of Real dataset (c) 33% of Real dataset + 0% of simulation data (d) 33% of Real dataset + 33% of simulation data (e) 33% of Real data + 66% of simulation data (f) 33% of Real data + 100% of simulation data.

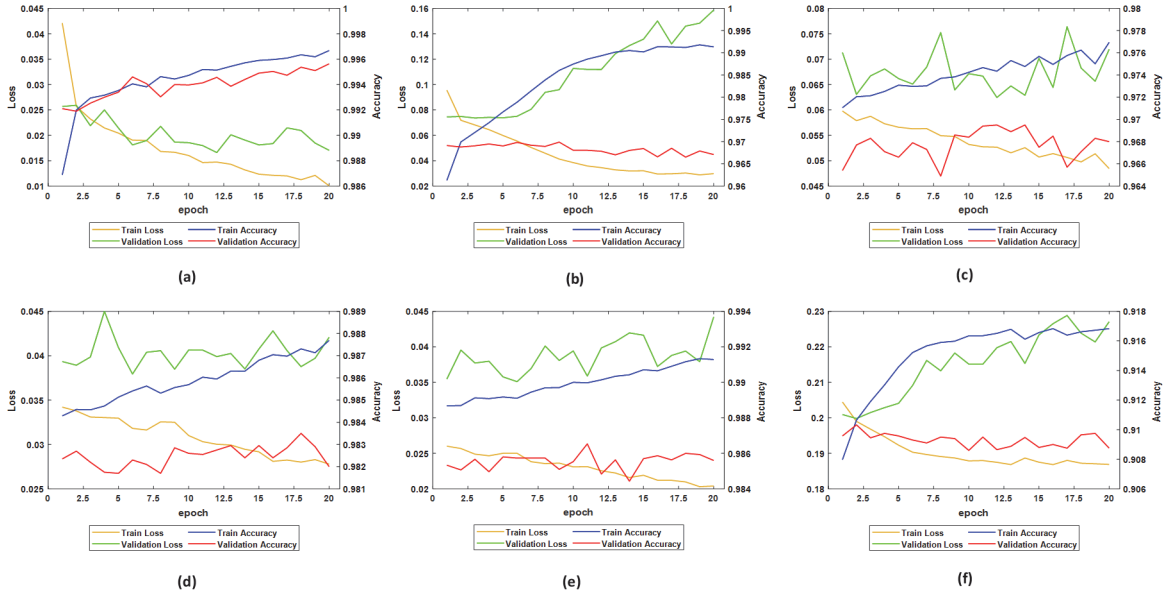


Figure 8: Training graphs of the CNN models trained from scratch (a) 100% of Simulation dataset (b) 100% of Real dataset (c) 33% of Real dataset + 0% of simulation data (d) 33% of Real dataset + 33% of simulation data (e) 33% of Real data + 66% of simulation data (f) 33% of Real data + 100% of simulation data.

CNN models where a, b, c, d, e, and f refers to the six dataset variations mentioned above.

Table 2: Training loss and accuracy of the chosen fine-tuned CNN models.

| Model | Epoch | Training | |
|---|---|---|---|
| | | Loss | Accuracy |
| a | 19 | 0.012 | 0.99 |
| b | 16 | 0.048 | 0.97 |
| c | 13 | 0.064 | 0.96 |
| d | 17 | 0.036 | 0.98 |
| e | 13 | 0.032 | 0.98 |
| f | 19 | 0.022 | 0.99 |

To avoid overfitting and improve the generalization of the trained models the epoch with minimum validation loss value is selected for inference. Tables 2, 3 illustrate the chosen epoch number and the corresponding training loss and accuracy for all trained models.

Table 3: Training loss and accuracy of the chosen CNN models trained from scratch.

| Model | Epoch | Training | |
|---|---|---|---|
| | | Loss | Accuracy |
| a | 9 | 0.225 | 0.89 |
| b | 3 | 0.068 | 0.97 |
| c | 12 | 0.053 | 0.97 |
| d | 6 | 0.032 | 0.98 |
| e | 6 | 0.025 | 0.99 |
| f | 2 | 0.19 | 0.91 |

# 5 EVALUATION OF THE TRAINED CNN MODELS

To investigate the learning ability of the models we computed the confusion matrix, accuracy, precision, recall, and F1 score for all trained models. The results showed that there is a significant increase in the accuracy per class as well as overall accuracy with the continuous addition of simulation data. In other words, the accuracy of the model trained with 33% real-world data +100% simulation data is higher than the model trained with only 33% of real data. Table 4 depicts the overall accuracy for models trained with datasets c, d, e, and f.

Table 4: Overall accuracy for models trained with datasets c,d,e, and f.

| Model | Overall Accuracy | | | |
|---|---|---|---|---|
| | c | d | e | f |
| Fine-tuned | 0.816 | 0.9 | 0.93 | 0.95 |
| From scratch | 0.77 | 0.52 | 0.77 | 0.93 |

Table 5 depicts the F1 score for all trained models. The F1 score measures the model accuracy by calculating the weighted average of precision and recall. An F1 score value of 1 is considered perfect, i.e. model predictions have low false positives and negatives.

Table 5: F1 score of the trained models.

| Model | Class | Dataset | | | | | |
|---|---|---|---|---|---|---|---|
| | | a | b | c | d | e | f |
| Fine-tuned | 1 | 0.98 | 0.84 | 0.76 | 0.87 | 0.91 | 0.93 |
| | 2 | 0.96 | 0.80 | 0.71 | 0.87 | 0.89 | 0.92 |
| | 3 | 0.97 | 0.85 | 0.78 | 0.87 | 0.91 | 0.94 |
| | 4 | 0.96 | 0.99 | 1.0 | 0.98 | 0.97 | 0.98 |
| From scratch | 1 | 0.99 | 0.79 | 0.71 | 0.5 | 0.71 | 0.91 |
| | 2 | 0.97 | 0.78 | 0.71 | 0.5 | 0.71 | 0.89 |
| | 3 | 0.97 | 0.80 | 0.72 | 0.3 | 0.72 | 0.92 |
| | 4 | 0.96 | 0.94 | 0.91 | 0.56 | 0.9 | 1.0 |

Figure 9 illustrates an example of the confusion matrix for the models fine-tuned with datasets c, d, e, and f. The confusion matrix diagonal elements represent the accuracy of correct predictions per class. The analysis of the confusion matrix for all trained models showed a major confusion between one class and its neighbor classes. This confusion, however, arises because of the small difference in steering angle ‘‘10°’’ between one class and another as well as the small field of view of the camera ‘‘78°’’ which decreases the size of the region of interest drastically.

To analyze the models' performance further the frozen inference graphs of the models trained with datasets a, b, c, d, e, and f are obtained and deployed on both environments, i.e. simulation and real environments. In the inference stage, the models take the incoming image frame and infer the required action, i.e. vehicle velocity and steering angle. Several test runs have been conducted in both environments. The results of these experiments demonstrated that the model trained only with real or simulation data works only in its respective environment. However, the models trained with mixed data, i.e. real and simulation data generalize perfectly to both environments especially when the number of incorporated simulation data samples is increased to 100%.

## 6 DISCUSSION

Datasets are a very crucial element in the training of deep convolution neural networks as it requires a massive amount of training data which in most cases is difficult to find. Simulators can provide the domain with an infinite amount of data samples, however, they do not have a perfect representation of the physical world which results in a huge gap between both environments. Most of the research in the robotics field employs either simulation or real data but not a mixture of both. In this study, we demonstrated that CNN models trained with a mixture of real and artificial data generalize to both real and simulation environments without model adaption. This holds for fine-tuned models as well as models trained from scratch. These findings are in accordance with findings reported by Bayraktar, Yigit, and Boyraz (2019). However, the authors did not consider training models from scratch as well as

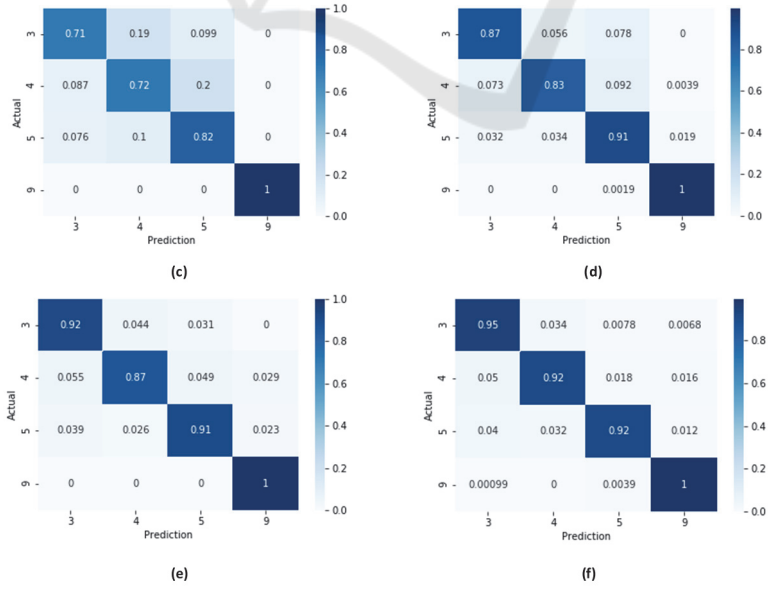


Figure 9: Fine-tuned CNN models confusion matrix (c) 33% of Real dataset + 0% of simulation data (d) 33% of Real dataset + 33% of simulation data (e) 33% of Real data + 66% of simulation data (f) 33% of Real data + 100% of simulation data

deploying the trained models to real and simulation environments.

Our results cast a new light on the performance and accuracy of the trained models. The models' overall accuracy and performance increased proportionally with the continuous addition of simulation data.

# 7 CONCLUSIONS AND FUTURE WORK

In this paper, we have investigated the ability of CNNs models to generalize and learn from a mixture of real and artificial data. Two CNN models have been evaluated where the first was trained from scratch and the other was fine-tuned to fit the dataset classes of the lane tracking task. The CNN models are trained six times wherein each a different combination of the collected real and simulation datasets is used. The results show that the models trained with a dataset collected from a particular environment can work only in this environment and fails when it is transferred to an unseen target environment. Another promising finding was that the models' performance increased significantly and were able to generalize to both real and simulation environments with the inclusion of simulation data. On this basis, we conclude that a mixture of simulation and real data can help the CNN models to generalize in cases where datasets are scarce and when models trained in a particular domain are transferred to an unseen target domain. This paper provides a good starting point for further research. In our future research, we intend to examine more complex model architectures and environments.

# REFERENCES


Bayraktar, E., Yigit, C., & Boyraz, P. (2019). A hybrid image dataset toward bridging the gap between real and simulation environments for robotics. (pp. 23–40). Machine Vision and Applications.

Bousmalis, K., & Levine, S. (2017, October 30). Closing the Simulation-to-Reality Gap for Deep Robotic Learning. Retrieved March 3, 2020, from https://ai.googleblog.com/2017/10/closing-simulation-to-reality-gap-for.html.

Cutler, M., Walsh, T. J., & How, J. P. (2014). Reinforcement learning with multi-fidelity simulators. (pp. 3888-3895). IEEE.

Duan, L., Xu, D., & Tsang, I. (2012). Learning with augmented features for heterogeneous domain adaptation. (pp. 667–674). arXiv.

Gamal, O., Imran, M., Roth, H., & Wahrburg, J. (2020). Assistive Parking Systems Knowledge Transfer to End-to-End Deep Learning for Autonomous Parking. (pp. 216-221). IEEE.

Hoffman, J., Guadarrama, S., Tzeng, E. S., Hu, R., Donahue, J., Girshick, R., Darrell, T., Saenko, K. (2014). LSDA: Large scale detection through adaptation. (pp. 3536-3544). arXiv.

James, S., Davison, A. J., & Johns, E. (2017). Transferring end-to-end visuomotor control from simulation to real world for a multi-stage task. arXiv.

Kulis, B., Saenko, K., & Darrell, T. (2011). What you saw is not what you get: Domain adaptation using asymmetric kernel transforms. (pp. 1785-1792). IEEE.

Li, Y., Wang, N., Shi, J., Liu, J., & Hou, X. (2016). Revisiting batch normalization for practical domain adaptation. arXiv.

Lin, M., Chen, Q., & Yan, S. (2013). Network in network. arXiv.

Loquercio, A., Kaufmann, E., Ranftl, R., Dosovitskiy, A., Koltun, V., & Scaramuzza, D. (2019). Deep drone racing: From simulation to reality with domain randomization. *36*. IEEE Transactions on Robotics.

Peng, X., Sun, B., Ali, K., & Saenko, K. (2014). Exploring invariances in deep convolutional neural networks using synthetic images. arXiv.

Sadeghi, F., & Levine, S. (2016). Cad2rl: Real single-image flight without a single real image. arXiv.

Stark, M., Goesele, M., & Schiele, B. (2010). Back to the Future: Learning Shape Models from 3D CAD Data. 2, p. 5. BMVC.

Su, H., Qi, C. R., Li, Y., & Guibas, L. J. (2015). Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views. (pp. 2686-2694). ICCV.

Sünderhauf, N., Brock, O., Scheirer, W., Hadsell, R., Fox, D., Leitner, J., Upcroft, B., Abbeel, P., Burgard, W., Milford, M., Corke, P. (2018). The limits and potentials of deep learning for robotics. *37*, pp. 405-420. IJRR.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A. (2015). Going deeper with convolutions. (pp. 1-9). IEEE.

Tkacz, M. (2005). Artificial neural networks in incomplete data sets processing. In M. Kłopotek, S. Wierzchoń, & T. K., *Intelligent Information Processing and Web Mining* (Vol. 31, pp. 577-583). Berlin, Heidelberg, Berlin: Springer.

Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., & Abbeel, P. (2017). Domain randomization for transferring deep neural networks from simulation to the real world. (pp. 23-30). IEEE.

Tzeng, E., Hoffman, J., Zhang, N., Saenko, K., & Darrell, T. (2014). Deep Domain Confusion: Maximizing for Domain Invariance. arXiv.

Yan, M., Frosio, I., Tyree, S., & Kautz, J. (2017). Sim-to-real transfer of accurate grasping with eye-in-hand observations and continuous control. arXiv.

Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). How transferable are features in deep neural networks? *NIPS '14*. NIPS Foundation.

Yu, Z., & Yali, W. (2011). Analyses on Influence of Training Data Set to Neural Network Supervised Learning Performance. In L. S. Jin D., *Advances in Computer Science, Intelligent System and Environment. Advances in Intelligent and Soft Computing* (Vol. 106). Heidelberg, Berlin: Springer.

Zhang, F., Leitner, J., Milford, M., & Corke, P. (2016). Modular deep q networks for sim-to-real transfer of visuo-motor policies. arXiv.