電機資訊學院 2021 作 BRAIN PLUS HAND 實作專題競賽

Visual Domain Transfer for Autonomous Driving Agent

(EECS01)

Bo-Wun Cheng, Chien Feng, Jia-Yin Foo, Yan-Bin Diau

Abstract

This project proposes a new framework that aims to enhance the performance of a self-driving deep reinforcement learning (DRL) agent that is deployed to a domain different from that it is trained. We propose a new semantics segmentation based unsupervised domain adaptation (UDA) model that bridges the gap between different domains, e.g., CARLA simulator to real-world, by incorporating depth information as the perception module of the agent. The performance of the network is evaluated on SYNTHIA to Cityscapes benchmark and two scenes in CARLA simulator that possess different weather and lighting conditions. With the help of the perception module, the DRL agent trained in different scenes can be deployed into unseen domains with minimal performance degradation. Furthermore, since an UDA approach is adopted by our perception module, the semantics segmentation and depth annotations of the target domain are not required, which makes transferring the agent to another domain much easier. To sum up, our work opens up opportunities for future works that attempt to train DRL agents in simulated environments and deploy them to the real world.

Contributions

(1) We propose a novel UDA method that incorporates depth information.
(2) We allow a driver agent to be deployed domains different to that it is trained into domains.
(3) We design a self-driving agent environment interface to train RL driving agent.

Framework Overview

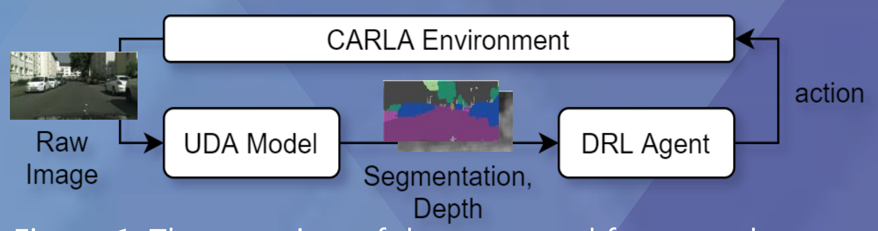

Figure 1. The overview of the proposed framework.

Our framework comprises two modules, the DRL self-driving agent, and a dual task perception model. During the training phase, the DRL agent is trained in a simulated environment and takes semantic segmentation and depth information generated by the environment as its visual input. An UDA model is trained separately with the image-label pairs in the source domain and images in the target domain. During the deployment phase, the UDA model generates the semantic and depth information in the target domain for the DRL agent as visual inputs. Given the visual input and other information, the self-driving DRL agent generates actions to accomplish the tasks it is trained to perform.

CARLA Environment

We use CARLA, an open-source driving simulator as the training environment for our RL model. CARLA is based on Unreal Engine to run the simulation. It serves as a modular API with a client-server architecture. The server is responsible for the simulation, while the client-side controls the logic of actors. Using CARLA' s Python API, we built an RL environment (CarlaEnv) on the client-side to meet the OpenAI Gym standards. The architecture is illustrated below:

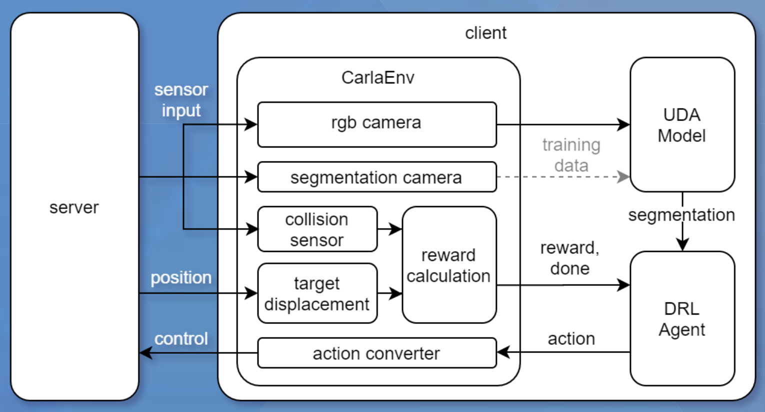

Figure 2. The overview of CARLA environment.

UDA Perception Model

The perception model of our framework is a dual task UDA model that aims to adapt from the source domain to the target domain by training on the image-label pairs in the source domain and the images in the target domain. The network consists of a generator and two discriminators and is trained in an adversarial fashion. The generator generates semantic segmentation and depth of the corresponding input image. The discriminator takes the output of the generator and tries to determine whether the original input is from the source domain or the target domain, and a discrimination loss is induced. If the input image is from the source domain, a loss is imposed such that the cross entropy loss between the generated output and the ground truth is minimized. As a result of training, our model successfully adapt to the target domain and outperform previous SOTA methods. The tables demonstrates the performance of our network measured in two scenarios in terms of mIoU.

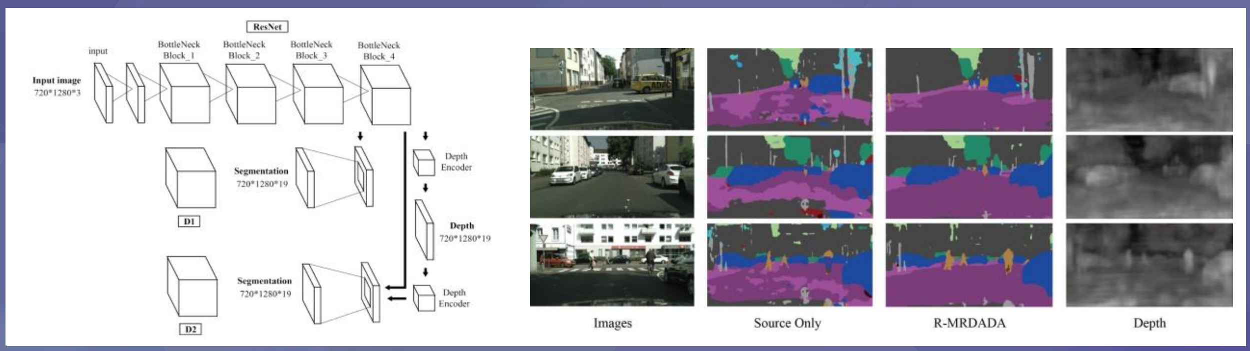

Figure 3. The overview of the perception model and the visualized results.

Model	Depth	Road	SideW	Build	Wall*	Fence*	Pole*	Light	Sign	Veg	Sky	Person	Rider	Car	Bus	Motor	Bike	mIOU	mIOU*
MRNet (Source only)		44.0	19.3	70.9	8.7	0.8	28.2	16.1	16.7	79.8	81.4	57.8	19.2	46.9	17.2	12.0	43.8	35.2	40.4
MRNet		82.0	36.5	80.4	4.2	0.4	33.7	18.0	13.4	81.1	80.8	61.3	21.7	84.4	32.4	14.8	45.7	43.2	50.2
R-MRNet		87.6	41.9	83.1	14.7	1.7	36.2	31.3	19.9	81.6	80.6	63.0	21.8	86.2	40.7	23.6	53.1	47.9	54.9
SPIGAN	V	71.1	29.8	71.4	3.7	0.3	33.2	6.4	15.6	81.2	78.9	52.7	13.1	75.9	25.5	10.0	20.5	36.8	42.4
DADA	V	89.2	44.8	81.4	6.8	0.3	26.2	8.6	11.1	81.8	84.0	54.7	19.3	79.7	40.7	14.0	38.8	42.6	49.8
MRDADA (Source only)	V	37.3	23.5	53.4	10.6	1.0	27.1	25.5	29.2	70.1	73.0	55.4	23.8	66.8	22.8	17.2	40.0	36.0	41.4
MRDADA	V	82.2	36.1	77.0	5.7	0.6	32.7	15.1	16.3	73.4	79.5	58.9	11.4	82.0	33.8	8.9	39.4	40.8	47.2
R-MRDADA	V	82.6	37.0	77.4	8.2	1.3	30.4	25.4	25.5	69.7	80.1	61.8	11.5	83.3	39.7	19.3	50.5	44.0	51.1

Table 1. Performance comparison with the other methods in SYNTHIA to Cityscapes.

Model	Depth	Road	SideW	Build	Wall	Fence	Pole	Sign	Veg	Person	Car	Line	mIOU
MRNet (Oracle)		93.1	90.3	93.4	81.0	60.8	63.9	42.5	90.1	54.4	81.6	47.1	72.6
MRNet (Source only)		39.2	49.8	75.2	26.1	13.3	36.3	18.0	79.4	46.7	67.9	6.6	41.7
MRNet		21.5	53.1	81.8	38.0	16.7	39.6	22.1	84.1	58.2	77.3	15.4	46.2
R-MRNet		25.8	53.1	82.8	39.8	17.6	36.8	23.5	84.3	55.6	79.9	21.3	47.3
MRDADA (Oracle)	V	92.1	89.5	93.6	82.6	63.7	61.7	43.6	90.6	54.9	82.8	41.9	72.5
MRDADA (Source only)	V	47.1	46.4	78.4	26.0	12.3	34.0	18.0	84.3	47.3	66.5	5.8	42.4
MRDADA	V	35.7	51.6	80.3	25.7	14.6	29.7	20.9	84.6	49.1	71.0	5.3	42.6
R-MRDADA	V	44.6	51.6	82.0	28.3	16.5	25.8	20.2	84.9	45.0	76.7	4.6	43.7

Table 2. Performance comparison with the other methods in CARLA (town1 to town3)

DRL Driving Agent

For the DRL agent, PPO is used as the RL algorithm because of its efficiency and stability. PPO have been extensively used in previous works, such as Unity ML Agent, Atari games, and other self-driving works, and have achieved remarkable results. However, different to previous works, we incorporate additional information to assist our DRL agent, including speed, steering, and displacement vector. Displacement vectors are widely used in self-driving tasks, which provide guidance for the agent to approach the goal.