Domain Adaptation for Imitation Learning

Rishabh Madan* rishabhm@iitkgp.ac.in
Rohit Gajawada* rohit.gajawada@students.iiit.ac.in
Antonio M. Lopez antonio@cvc.uab.es

1. Introduction

Deep networks trained on real world data by using imitation learning have shown to produce very good results. These end to end imitation learning approaches are very powerful and perform quite well on the same domain where the data was collected. However, the performance of an imitation learning agent is highly dependent on the amount and quality of data that it is trained on. Collecting a large amount of high quality data from the real world is not a trivial task and usually involves a complicated setup. With the advent of simulators like TORCS and CARLA, we now have a massive amount of data. The problem is that agents trained on simulated data perform terrible in the real world due to the enormous domain shift. Solving this problem of domain adaptation for imitation learning agents is very crucial in order to improve the performance of these agents in real world settings. In this report, we will discuss about the current problems of domain adaptation for imitation learning, existing works that are able to solve some of the problems and some experiments that we have run.

2. Imitation Learning

Traditionally, there has been a practice of using a modular pipeline involving perception, localization, path search and controls modules for the motion planning of an autonomous vehicle. This approach makes use of numerous sensor data to drive a car. In contrast, people have also explored end to end driving approaches with minimal number of sensors using methods like imitation learning. One of the earliest works ALVINN [13], involves using a neural network to predict driver controls by just taking the RGB camera frames as input. Since then, several techniques have been introduced involving imitation learning that have shown competent results. Deep networks trained on demonstrations of human driving have shown to learn to follow roads and avoid obstacles. There has also been recent work involving command-conditional formulation that enables the application of imitation learning to more complex urban driving. If the training data includes observations of recoveries from perturbations, even very small deep imitation learning agents perform very well on that particular domain. However even though if these agents perform very well on domains similar to the domain on which they are train, they fail miserably when generalizing to other domains. Agents trained on simulator like CARLA fail miserably when tested on different weathers or in a real world setup.

*These authors have contributed equally

3. Domain Adaptation

Domain adaptation is a section of machine learning, that addresses the issue of domain shift between two data distributions and makes use of labeled data in source domain to learn the task in hand, for the target domain even if target domain labeled data is unavailable. Methods using deep learning for Domain adaptation have shown superior results in the recent past. Early deep adaptive works focused on feature space alignment through minimizing the distance between first or second order feature space statistics of the source and target ([19]; [11]). Other related techniques involve learning a mapping from one domain to the other at a feature level. In such a setup, the feature extraction pipeline is fixed during the domain adaptation optimization. This has been applied in various non-CNN based approaches [6], [2], [5][2, 3, 4] as well as CNN-based Correlation Alignment (CORAL) [16] algorithm. Tobin et al in [18] make use of domain randomization in simulation to transfer the task of robotic grasping to realistic objects.

3.1. Using Adversarial approaches

3.1.1 Utilizing the Pixel Space

PixelDA [1] decouples the process of domain adaptation from the task-specific architecture and performs unsupervised domain adaptation.

CycleGAN (Zhu et al., 2017) [24] produced compelling image translation results such as generating photorealistic images from impressionism paintings or transforming horses into zebras at high resolution using the cycle-consistency loss. However, its performance suffered for domain adaptation as it tends to hallucinate objects and loses semantic information.

[11] uses style transfer and content loss along with cycle consistency loss to improve domain adaptation for monocu-

lar depth estimation. The approach doesn't work well when adapting to sudden lighting changes and saturation during style transfer.

3.1.2 Utilizing the Feature Space

The Domain Transfer Network (Taigman et al., 2017b) [17] trains a generator to transform a source image into a target image by enforcing consistency in the embedding space. Shrivastava et al. (2017) [15] instead use an L1 reconstruction loss to force the generated target images to be similar to their original source images. This works well for limited domain shifts where the domains are similar in pixel-space, but can be too limiting for settings with larger domain shifts.

BiGAN (Donahue et al., 2017) [3] and ALI (Dumoulin et al., 2016) [4] take an approach of simultaneously learning the transformations between the pixel and the latent space.

CoGANs (Liu & Tuzel, 2016b) [10] jointly learn a source and target representation through explicit weight sharing of certain layers while each source and target has a unique generative adversarial objective.

WDGRL [9] [10] utilizes a discriminator to estimate empirical Wasserstein distance between the source and target samples and optimizes the feature extractor network to minimize the estimated Wasserstein distance.

[10] [20] forces the learned feature extractor to be domain-invariant, and training it through data augmentation in the feature space, namely performing feature augmentation.

UNIT [9]makes a shared-latent space assumption and relies on both GANs and Variational Auto-Encoders, which assumes a pair of corresponding images in different domains can be mapped to a same latent representation in a shared-latent space. MUNIT [8] assumes that a pair of corresponding images in different domains can be mapped to a same content representation with different style representations.

CyCADA [7] adapts representations at both the pixel-level and feature-level, enforces cycle-consistency while leveraging a task loss, and does not require aligned pairs similar to CycleGAN.

LSDSEG [14] proposes a joint adversarial approach that transfers the information of the target distribution to the learned embedding using a generator-discriminator pair and significantly improves results for domain adaptation for semantic segmentation.

4. Domain Adaptation for Imitation Learning Agents

There have been driving datasets by Comma.ai [6] and Udacity [7] for self driving cars, that mostly consist of driving data for highways and roads in USA. IL agents trained on these datasets don't perform (might not perform) well

Figure 1. Top view of the Sant Quirze circuit in CARLA Simulator

in unseen environments and conditions. This is due to the lack of diverse training data for driving in terms of different roads, weather conditions and traffic conditions. With the advent of high quality simulators for autonomous driving, there is abundant synthetic data consisting of a wide variety of conditions when driving. There have been recent works that have shown impressive results when using imitation learning agents to drive a vehicle in a simulator. How do we make use of these agents to drive vehicles in real world? We use domain adaptation to make the agents adaptable to target domain i.e. the real world.

[23] uses a CycleGAN based architecture with style loss and utilize the target domain discriminator features to learn the task of driving.

[22] trains a generator discriminator pair with a Task network to transform real world images to virtual and learn the control task from the transformed images. The generator network is updated using both adversarial and prediction objective.

[12] transfer driving policy from virtual to real environment by training a encoder-decoder based segmentation architecture with a driving policy network that outputs waypoints, and use a low-level controller to drive the car.

[21] introduces a master-servant architecture, where the master model (semantic labels available) trains the servant model (semantic labels not available). The servant model

is then used for steering the vehicle without retraining the control module. However, this would not work for virtual to real domain adaptation as the segmentation network trained on virtual data would doesn't perform well on real world data.

We address the issue of domain adaptation for IL agents by making use of adversarial networks for learning a common feature space and use a task network to predict the controls for our vehicle.

5. Our Experiments

5.1. Training Environment

We performed all the experiments in the Sant Quirze Circuit environment (Fig. 1) of CARLA. The circuit comprises of a total of 6 turning points, most of them being complete U-Turns.

5.2. Training Imitation Learning Agent on CARLA

We train an IL agent similar to CIL with only branch instead, using the SantQuirze Circuit environment on CARLA. We train the agent on Low Quality Weather 1 (Clear Noon). Our domain adaptation objective is to perform well on Epic Quality Weather 12 (Mid Rain Sunset) which is the hardest setting due to rain, low illumination, heavy sunlight reflection from road and grass surfaces, etc. Also, Low Quality data is very simple and synthetic whereas Epic Quality looks very realistic. Our IL agents are trained for 200k iterations.

5.3. Finetuning agent on fake target domain images

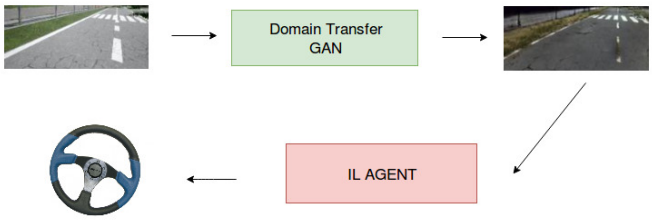

Figure 2. Finetuning agent on fake target domain images

We train the UNIT GAN to transfer images from source domain to target images for 100 epochs. After training, we take the target encoder and source decoder of the UNIT and convert the entire source dataset to a artificial target dataset by passing all source images through the UNIT.

After this, we take a pretrained IL agent from the source domain and finetune it on this artificial target dataset with a lower learning rate for 200k iterations.

5.4. Training with a joint adversarial approach

This methods aims to learn a common embedding and weights for both domain images. The input to this setup is a triplet sample consisting of a source domain image, the corresponding source domain action label and a random target domain image. It comprises of one encoder which embeds image into a latent space. This latent vector is passed through the common decoder to reconstruct the image. The fake reconstructed image is passed through a 4 label discriminator namely: Source Fake, Source Real, Target Fake and Target Real and the discriminator must learn to discriminate between the 4 possibilities. The latent space availed from the encoder is directly passed into the task network and the task is learnt on the source data. In theory, since the target images are passed through the same encoder and still need to be reconstructed from that latent space, domain adaptation for the driving task should be performed. However, we found the entire setup to be very unstable as the discriminator was getting confused between source fake and target fake and ended up causing the image reconstructions to mode collapse. Even using pretrained weights for the encoder (IL Agent weights) and decoder (WGAN-GP reconstruction or L1 reconstruction) did not fix the issue. Our training procedure for this experiment as well as the next one is a 3 step process.

First, the discriminator is updated with adversarial loss for distinguishing between fake and real samples.

Second, the generator (only decoder) is updated with adversarial and L1 loss to make fake images realistic (Source Fake to Source Real and Target Fake to Target Real).

Finally, the generator (only encoder) and task network are updated with two components. One component is the task loss. The second component is the adversarial loss (Source Fake to Target Real and Target Fake to Source Real).

$$\min_G \max_D V(D,G) = \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_z(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))]$$

$$\text{Task Loss} = \frac{1}{n}\sum_{t=1}^{n}(pred_t - y_t)^2$$

where $pred_t$ is of the form: [steer, throttle, brake]

5.5. Training with a joint adversarial approach with extra WGAN-GP stabilization

To address the problem of mode collapse and unstable training, we add a separate discriminator trained using WGAN-GP loss for the concept of fake and real. However, having two discriminators: one with 4 labels and GAN loss and another with WGAN-GP loss was causing the WGAN-

Source Domain	Tested Domain	Results
Low W1	Low W1	Perfect
Low W1	Low W1 w/o lanes	Perfect
Low W1	Epic W1	All turns except heavy shadow
Low W1	Epic W1 w/o lanes	All turns except heavy shadow
Low W1	Epic W12	Does horribly but makes few turns
Low W1	Epic W12 w/o lanes	Immediately strays off into grass
Epic W12	Epic W12	Works perfect but lane follower
Epic W12	Epic W12 w/o lanes	Unable to follow road and strays off

Table 1. Results for CIL agents on specified domains

Method	Source Domain	Target Domain
UNIT Finetune	Low W1	Epic W12 no lanes
LSDSEG based with WGAN	Low W1	Epic W12 no lanes
WDGRL	Low W1	Epic W12 no lanes
UNIT Latent Space DA	Low W1	Epic W12 no lanes

Table 2. Experiments conducted

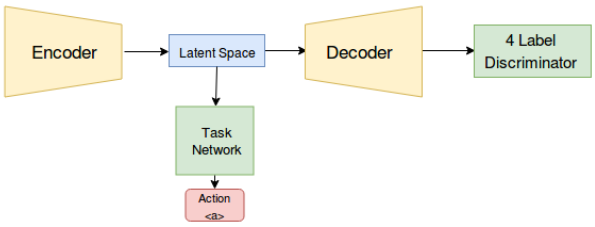

Figure 3. Training with a joint adversarial approach

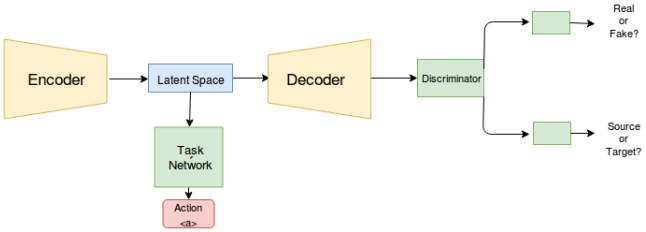

Figure 4. Training with a joint adversarial approach with shared weights

GP loss to override even though if the setup was stable and image reconstructions were good.

The final setup consisted of one discriminator but performed two tasks. One task was to classify if the image was real or fake and the other task was to classify if the image was from source domain or target domain. Both tasks have shared weights until the last two layers. This gave us better results.

This architecture is trained similar to our joint adversarial approach in the previous subsection.

$$L = \mathbb{E}_{\tilde{\mathbf{x}}\sim\mathbb{P}_g}\left[D(\tilde{\mathbf{x}})\right] - \mathbb{E}_{\mathbf{x}\sim\mathbb{P}_r}\left[D(\mathbf{x})\right] + \lambda \mathbb{E}_{\hat{\mathbf{x}}\sim\mathbb{P}}\left[\left(\|\nabla_{\hat{\mathbf{x}}} D(\hat{\mathbf{x}})\|_2 - 1\right)^2\right]$$

5.6. Adversarial feature adaptation based on WDGRL

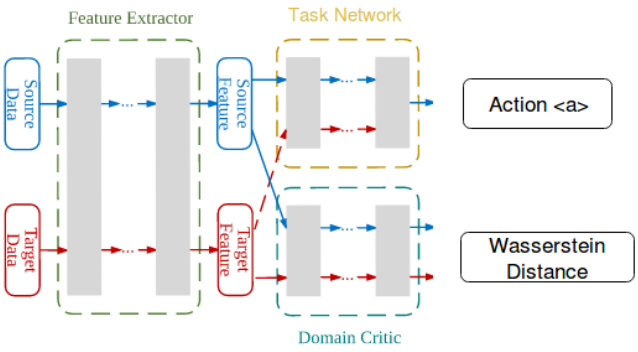

Figure 5. Adversarial feature adaptation based on WDGRL

We use the idea of WDGRL to use an external discriminator to discriminate between features of a source image and the features of a target images. Images from both domains are passed through the feature extractor but only task loss is applied on features from source images. The discriminator learns to discriminate between both feature which allows the feature extractor to learn to create a common feature space for both domain images.

5.7. Training from common latent space of UNITGAN

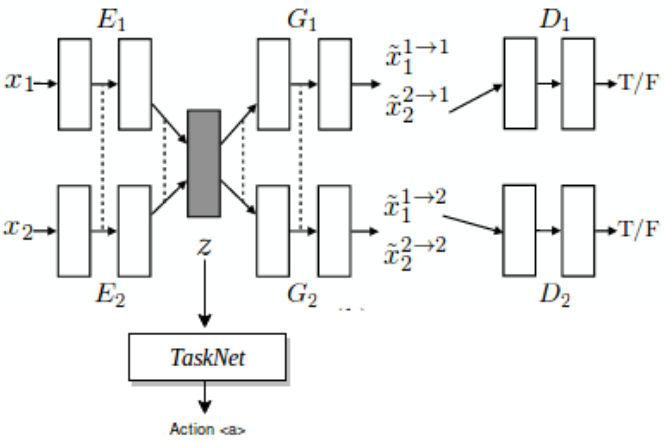

Figure 6. Training from common latent space of UNITGAN

This latent space of UNIT is domain invariant and contains common content for both the source domain images as well as the target domain image. We add a task network to the latent space of UNIT and train all networks end to end from scratch. Th goal is to learn the latent space that is informative enough to be able to regenerate the actual scene and at the same time contain information that is relevant for robust decision-making agent by the TaskNet. We vary the values of task weight and KL divergence and found KL divergence of 0.1 and task weight of 0.01 works best for our setting.

The UNIT GAN loss function is as follows:

$$\min_{E_1,E_2,G_2,G_2} \max_{D_1,D_2} \mathcal{L}_{\text{VAE}_1}(E_1,G_1) + \mathcal{L}_{\text{GAN}_1}(E_2,G_1,D_1) + \mathcal{L}_{\text{CC}_1}(E_1,G_1,E_2,G_2) + \mathcal{L}_{\text{VAE}_2}(E_2,G_2) + \mathcal{L}_{\text{GAN}_2}(E_1,G_2,D_2) + \mathcal{L}_{\text{CC}_2}(E_2,G_2,E_1,G_1)$$

$$\mathcal{L}_{\text{VAE}_1}(E_1,G_1) = \lambda_1 \text{KL}(q_1(z_1|x_1)\,||p_\eta(z)) - \lambda_2 \mathbb{E}_{z_1\sim q_1}(z_1|x_1)\,[\log p_{G_1}(x_1|z_1)]$$

$$\mathcal{L}_{\text{VAE}_2}(E_2,G_2) = \lambda_1 \text{KL}(q_2(z_2|x_2)\,||p_\eta(z)) - \lambda_2 \mathbb{E}_{z_2\sim q_2}(z_2|x_2)\,[\log p_{G_2}(x_2|z_2))$$

$$\mathcal{L}_{\text{GAN}_1}(E_2,G_1,D_1) = \lambda_0 \mathbb{E}_{x_1\sim P_{X_1}}[\log D_1(x_1)] + \lambda_0 \mathbb{E}_{z_2\sim q_2}(z_2|x_2)\,[\log(1-D_1(G_1(z_2)))]$$

$$\mathcal{L}_{\text{GAN}_2}(E_1,G_2,D_2) = \lambda_0 \mathbb{E}_{x_2\sim P_{X_2}}[\log D_2(x_2)] + \lambda_0 \mathbb{E}_{z_1\sim q_1(z_1|x_1)}[\log(1-D_2(G_2(z_1)))]$$

$$\mathcal{L}_{\text{CC}_1}(E_1,G_1,E_2,G_2) = \lambda_3 \text{KL}(q_1(z_1|x_1)\,||p_\eta(z)) + \lambda_3 \text{KL}(q_2|x_1^{1\to2}))||p_\eta(z)) - \lambda_4 \mathbb{E}_{z_2\sim q_2}(z_2|x_1^{1\to2})\,[\log p_{G_1}(x_1|z_2)]$$

$$\mathcal{L}_{\text{CC}_2}(E_2,G_2,E_1,G_1) = \lambda_3 \text{KL}(q_2|x_2)\,||p_\eta(z)) + \lambda_3 \text{KL}(q_1|x_2^{2\to1}))||p_\eta(z)) - \lambda_4 \mathbb{E}_{z_1\sim q_1}(z_1|x_2^{2\to1})\,[\log p_{G_2}(x_2|z_1)]$$

5.8. Implementation Details

All of our work was done in PyTorch. We use Adam optimizer for all of our experiments with $\beta_1 = 0.5$ and $\beta_2 = 0.99$. We collect data for our experiments using a setup of 5 cameras with angles -30°, -15°, 0°, 15°and 30°in CARLA simulator. For our UNIT Latent Space DA experiments we using input images of size 128 x 128 and for remaining we use 88 x 200 similar to CIL Felipe. Suitable steering augmentation values are added for cameras with non zero angle.

For all sections, we tried out several experiments involving the hyper-parameters of L1 weight, KL divergence, model size, learning rates, whether to use pretrained weights or not, etc.

5.9. Future Experiments

- Disentangling content from style in UNIT GAN like architecture task loss on content embedding and style loss on style embedding.
- Using good IL agents as encoders for UNIT Latent Space Domain Adaptation similar to our other experiments.
- Trying out all of our experiments on Town 01 and Town 02 and for many more easier weathers.
- Look into suitable pretrained weights for networks if required.
- Lifelong learning across several weathers to get best universal model.

References

[1] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. *CoRR*.

[2] K. Bousmalis, G. Trigeorgis, N. Silberman, D. Krishnan, and D. Erhan. Domain separation networks. NIPS'16, 2016.

[3] J. Donahue, P. Krähenbühl, and T. Darrell. Adversarial feature learning. *CoRR*, abs/1605.09782, 2016.

[4] V. Dumoulin, I. Belghazi, B. Poole, O. Mastropietro, A. Lamb, M. Arjovsky, and A. Courville. Adversarially Learned Inference. *ArXiv e-prints*, 2016.

[5] R. Gopalan, R. Li, and R. Chellappa. Domain adaptation for object recognition: An unsupervised approach. 2011.

[6] K. Grauman. Geodesic flow kernel for unsupervised domain adaptation. *CVPR*, 2012.

[7] J. Hoffman, E. Tzeng, T. Park, J. Zhu, P. Isola, K. Saenko, A. A. Efros, and T. Darrell. Cycada: Cycle-consistent adversarial domain adaptation. *CoRR*, abs/1711.03213, 2017.

[8] X. Huang, M. Liu, S. J. Belongie, and J. Kautz. Multimodal unsupervised image-to-image translation. *CoRR*, abs/1804.04732, 2018.

[9] M. Liu, T. Breuel, and J. Kautz. Unsupervised image-to-image translation networks. *CoRR*, abs/1703.00848, 2017.

[10] M.-Y. Liu and O. Tuzel. Coupled generative adversarial networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 469–477. 2016.

[11] M. Long and J. Wang. Learning transferable features with deep adaptation networks. *CoRR*, abs/1502.02791, 2015.

[12] M. Müller, A. Dosovitskiy, B. Ghanem, and V. Koltun. Driving policy transfer via modularity and abstraction. *CoRR*, abs/1804.09364, 2018.

[13] D. A. Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems 1*, pages 305–313. Morgan-Kaufmann, 1989.

[14] S. Sankaranarayanan, Y. Balaji, A. Jain, S. Lim, and R. Chellappa. Unsupervised domain adaptation for semantic segmentation with gans. *CoRR*.

[15] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb. Learning from simulated and unsupervised images through adversarial training. *CoRR*, 2016.

[16] B. Sun, J. Feng, and K. Saenko. Return of frustratingly easy domain adaptation. In *AAAI*, 2016.

[17] Y. Taigman, A. Polyak, and L. Wolf. Unsupervised cross-domain image generation. *CoRR*, abs/1611.02200, 2016.

[18] J. Tobin, W. Zaremba, and P. Abbeel. Domain randomization and generative models for robotic grasping. *CoRR*, abs/1710.06425, 2017.

[19] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell. Deep domain confusion: Maximizing for domain invariance. *CoRR*, abs/1412.3474, 2014.

[20] R. Volpi, P. Morerio, S. Savarese, and V. Murino. Adversarial feature augmentation for unsupervised domain adaptation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[21] P. Wenzel, Q. Khan, D. Cremers, and L. Leal-Taixé. Modular Vehicle Control for Transferring Semantic Information to Unseen Weather Conditions using GANs. *ArXiv e-prints*.

[22] L. Yang, X. Liang, and E. P. Xing. Unsupervised real-to-virtual domain unification for end-to-end highway driving. *CoRR*, abs/1801.03458, 2018.

[23] J. Yoo, Y. Hong, and S. Yoon. Autonomous UAV navigation with domain adaptation. *CoRR*.

[24] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, 2017.