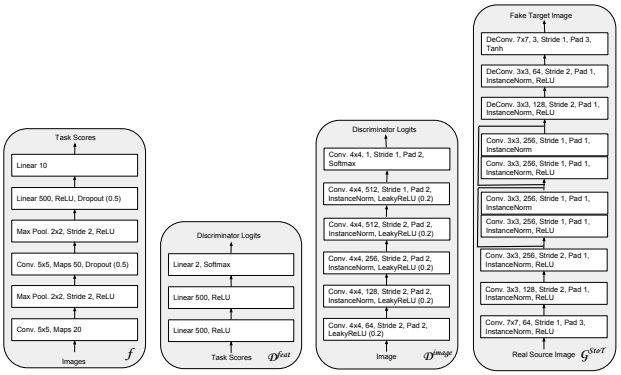


Figure 7: Network architectures used for digit experiments. We show here the task net (f), discriminator for feature level adaptation (D^{feat}), discriminator for image level adaptation (D^{image}), and generator for source to target (G) – same network used for target to source.

A. Appendix

A.1. Implementation Details

We begin by pretraining the source task model, f_S, using the task loss on the labeled source data. Next, we perform pixel-level adaptation using our image space GAN losses together with semantic consistency and cycle consistency losses. This yeilds learned parameters for the image transformations, $G_{S\to T}$ and $G_{T\to S}$, image discriminators, D_S and D_T, as well as an initial setting of the task model, f_T, which is trained using pixel transformed source images and the corresponding source pixel labels. Finally, we perform feature space adpatation in order to update the target semantic model, f_T, to have features which are aligned between the source images mapped into target style and the real target images. During this phase, we learn the feature discriminator, D_{feat} and use this to guide the representation update to f_T. In general, our method could also perform phases 2 and 3 simultaneously, but this would require more GPU memory then available at the time of these experiments.

For all feature space adaptation we equally weight the generator and discriminator losses. We only update the generator when the discriminator accuracy is above 60% over the last batch (digits) or last 100 iterations (semantic segmentation) – this reduces the potential for volatile training. If after an epoch (entire pass over dataset) no suitable discriminator is found, the feature adaptation stops, otherwise it continues until max iterations are reached.

A.1.1. Digit Experiments

For all digit experiments we use a variant of the LeNet architecture as the task net (Figure 7 *left*). Our feature discriminator network consists of 3 fully connected layers (Figure 7 *mid left*). The image discriminator network consists of 6 convolutional layers culminating in a single value per pixel (Figure 7 *mid right*). Finally, to generate one image domain from another we use a multilayer network which consists of convolution layers followed by two residual blocks and then deconvolution layers (Figure 7 *right*). All stages are trained using the Adam optimizer.

Hyperparameters. For training the source task net model, we use learning rate 1e-4 and train for 100 epochs over the data with batch size 128. For feature space adaptation we use learning rate 1e-5 and train for max 200 epochs over the data. For pixel space adaptation we train our generators and discriminators with equal weighting on all losses, use batch size 100, learning rate 2e-4 (default from CycleGAN), and trained for 50 epochs. We ran each experiment 4 times and report the average and standard error across the runs.

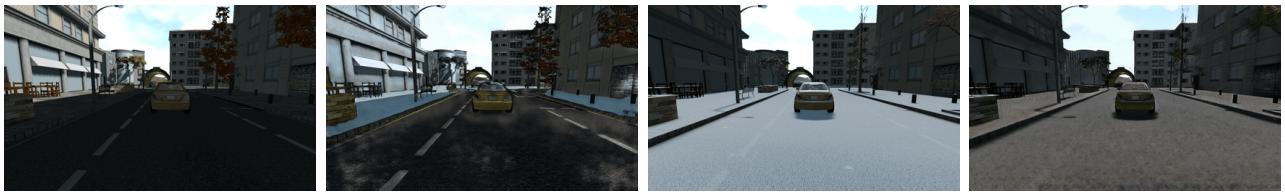
A.1.2. Semantic Segmentation

We experiment with both the VGG16-FCN8s (Long et al., 2015) architecture as well as the DRN-26 (Yu et al., 2017) architecture. For FCN8s, we train our source semantic segmentation model for 100k iterations using SGD with learning rate 1e-3 and momentum 0.9. For the DRN-26 architecture, we train our source semantic segmentation model for 115K iterations using SGD with learning rate 1e-3 and momentum 0.9. We use a crop size of 600x600 and a batch size of 8 for this training. For cycle-consistent image level adaptation, we followed the network architecture and hyperparameters of CycleGAN(Zhu et al., 2017). All images were resized to have width of 1024 pixels while keeping the aspect ratio, and the training was performed with randomly cropped patches of size 400 by 400. Also, due to large size of the dataset, we trained only 20 epochs. For feature level adaptation, we train using SGD with momentum, 0.99, and learning rate 1e-5. We weight the representation loss ten times less than the discriminator loss as a convenience since otherwise the discriminator did not learn a suitable model within a single epoch. Then the segmentation model was trained separately using the adapted source images and the ground truth labels of the source data. Due to memory limitations we can only include a single source and single target image at a time (crops of size 768x768), this small batch is one of the main reasons for using a high momentum parameter.

A.2. Cross-season adaptation

As an additional semantic segmentation evaluation, we consider the SYNTHIA dataset (Ros et al., 2016a), which contains synthetic renderings of urban scenes. We use the SYNTHIA video sequences, which are rendered across a variety of environments, weather conditions, and lighting conditions. This provides a synthetic testbed for evaluating adaptation techniques. For comparison with previous work, in this work we focus on adaptation between seasons. We use only the front-facing views in the sequences so as to mimic dashcam imagery, and adapt from fall to winter. The subset of the dataset we use contains 13 classes and consists of 10,852 fall images and 7,654 winter images.

We start by exploring the abilities of pixel space adaptation alone (using FCN8s architecture) for the setting of adapting across seasons in synthetic data. For this we use the SYNTHIA dataset and adapt from fall to winter weather conditions. Typically in unsupervised adaptation settings it is difficult to interpret what causes the performance improvement after adaptation. Therefore, we use this setting as an example where we may directly visualize the shift from fall to winter and inspect the intermediate pixel level adaptation result from our algorithm. In Figure 8 we show the result of pixel only adaptation as we generate a winter domain image (b) from a fall domain image (a), and visa versa (c-d). We may clearly see the changes of adding or removing snow. This visually interpretable result matches our expectation of the true shift between these domains and indeed results in favorable final semantic segmentation performance from fall to winter as shown in Table 6.



(a) Fall (b) Fall → Winter (c) Winter (d) Winter → Fall

Figure 8: **Cross Season Image Translation.** Example image-space conversions for the SYNTHIA seasons adaptation setting. We show real samples from each domain (Fall and Winter) alongside conversions to the opposite domain.

SYNTHIA Fall → Winter																
	sky	building	road	sidewalk	fence	vegetation	pole	car	traffic sign	pedestrian	bicycle	lanemarking	traffic light	**mIoU**	**fwIoU**	**Pixel acc.**
Source only	91.7	80.6	79.7	12.1	71.8	44.2	26.1	42.8	49.0	38.7	45.1	41.3	24.5	49.8	71.7	82.3
FCNs in the wild	92.1	86.7	91.3	20.8	72.7	**52.9**	**46.5**	64.3	**50.0**	**59.5**	**54.6**	**57.5**	**26.1**	59.6	—	—
CyCADA pixel-only	**92.5**	**90.1**	**91.9**	**79.9**	**85.7**	47.1	36.9	**82.6**	45.0	49.1	46.2	54.6	21.5	**63.3**	**85.7**	**92.1**
Oracle (Train on target)	93.8	92.2	94.7	90.7	90.2	64.4	38.1	88.5	55.4	51.0	52.0	68.9	37.3	70.5	89.9	94.5

Table 6: Adaptation between seasons in the SYNTHIA dataset. We report IoU for each class and mean IoU, freq-weighted IoU and pixel accuracy. Our CyCADA method achieves state-of-the-art performance on average across all categories. *FCNs in the wild is by Hoffman et al. (2016).

We find that CyCADA achieves state-of-the-art performance on this task with image space adaptation alone, however does not recover full supervised learning performance (train on target). Some example errors includes adding snow to the sidewalks, but not to the road, while in the true winter domain snow appears in both locations. However, even this mistake is interesting as it implies that the model is learning to distinguish road from sidewalk during pixel adaptation, despite the lack of pixel annotations.

Cycle-consistent adversarial adaptation achieves state-of-the-art adaptation performance. We see that under the fwIoU and pixel accuracy metrics, CyCADA approaches oracle performance, falling short by only a few points, despite being entirely unsupervised. This indicates that CyCADA is extremely effective at correcting the most common classes in the dataset. This conclusion is supported by inspection of the individual classes in Table 6, where we see the largest improvement on common classes such as *road* and *sidewalk*.

A.3. Comparison to Shrivastava et al. (2017) for Semantic Segmentation

We illustrate the performance of a recent pixel level adaptation approach proposed by Shrivastava et al. (2017) on our semantic segmentation data – GTA to Cityscapes. These images are significantly larger and more complex than those shown in the experiments in the original paper. We show image to image translation results under three different settings of the model hyperparameter, λ, which controls the tradeoff between the reconstruction loss and the visual style loss. When $\lambda = 10$ (Figure 9 *right*), the resulting image converges to a near replica of the original image, thus preserving content but lacking the correct target style. When $\lambda = 1$ or $\lambda = 2.5$ (Figure 9 *left*), the results lack any consistent semantics making it difficult to perceive the style of the transformed image. Thus, the resulting performance for this model is 11.6 mIoU for FCN8s with VGG, well below the performance of the corresponding source model of 17.9 mIoU.

(a) $\lambda = 1$ (b) $\lambda = 2.5$ (c) $\lambda = 10$

Figure 9: Image transformation results from Shrivastava et al. (2017) applied to GTA to CityScapes transformation. We demonstrate results using three different settings for λ.

A.4. Additional GTA to CityScapes Visualizations

We show additional image-space adaptation visualiztions for the GTA to CityScapes scenario in Figure 10.

(a) GTA5 (b) GTA5 → Cityscapes (c) CityScapes (d) CityScapes → GTA5

Figure 10: **GTA5 to CityScapes Image Translation.** Example images from the GTA5 (a) and Cityscapes (c) datasets, alongside their image-space conversions to the opposite domain, (b) and (d), respectively. Our model achieves highly realistic domain conversions.