Supplementary Material
From source to target and back: Symmetric Bi-Directional Adaptive GAN

Paolo Russo[1,2], Fabio M. Carlucci[1,2], Tatiana Tommasi[2] and Barbara Caputo[1,2]
[1]Department DIAG, Sapienza University of Rome, Italy
[2]Italian Institute of Technology
{Paolo.Russo, Fabio.Carlucci, Tatiana.Tommasi, Barbara.Caputo}@iit.it

1. SBADA-GAN network architecture

We composed SBADA-GAN starting from two symmetric GANs, each with an architecture analogous to that used for the PixelDA model. Specifically:

- the generators take the form of a convolutional residual network with four residual blocks each composed by two convolutional layers with 64 features;
- the input noise $\boldsymbol{z}$ is a vector of N^z elements each sampled from a normal distribution $z_i \sim \mathcal{N}(0, 1)$. It is fed to a fully connected layer which transforms it to a channel of the same resolution as that of the image, and is subsequently concatenated to the input as an extra channel. In all our experiments we used $N^z = 5$;
- the discriminators are made of two convolutional layers, followed by an average pooling and a convolution that brings the discriminator output to a single scalar value;
- in both generator and discriminator networks, each convolution (with the exception of the last one of the generator) is followed by a batch norm layer [4];
- the classifiers have exactly the same structure of that in [1, 2];
- as activation functions we used ReLU in the generator and classifier, while we used leaky ReLU (with a 0.2 slope) in the discriminator;
- all the input images to the generators are zero-centered and rescaled to $[-0.5, 0.5]$. The images produced by the generators as well as the other input images to the classifiers and and the discriminators are zero-centered and rescaled to $[-127.5, 127.5]$.

Thanks to the stability of the SBADA-GAN training protocol, we did not use any injected noise into the discriminators and we did not use any dropout layer.

2. More Implementation Details

The training batch: with "batch size = 32" we mean that 32 samples are randomly chosen from the source as well as from the target. The model works fine with different batch size values (*e.g.* 16,64).

Update policy for $y^j_{t_{self}}$ and C_s: the classifier C_s is first trained using only source images. After convergence, it is used to produce $y^j_{t_{self}}$.The target images annotated with these pseudo-labels contribute as input to C_s as follows (iterated): the associated self-labeling loss is used to provide feedback to G_{ts}, but it does not contribute to the update of C_s. Similarly the class consistency loss does not contribute to the update of C_s, but only to that of G_{st} and G_{ts}. $y^j_{t_{self}}$ can change as C_s is still trained at each iteration.

C_s,C_t **combination:** the performance of C_t is already good and better than several baselines. The improvement provided by C_s is evident even when the two classifiers are integrated with fixed weights (*e.g.* $\sigma = 0.3$ or 0.5), thus a detailed search for the weights values is not strictly necessary. Anyway we did it by exploiting only a subset of the target samples, as [4] did to select their model parameters.

3. Self-Labeling

Self-labeling may appear as an unsafe procedure in case of large domain shift between source and target. To understand the low risk provided by self-labeling in SBADAGAN we remark that both the classifiers C_s and C_t are trained on source images with ground truth labels. G_{st} is influenced and regularized by both these classifiers so that it is highly unlikely that a source image is deformed and appears as belonging to a different category. G_{ts} is slightly weaker as it deals with unlabeled target images, but it is helped by the class consistency loss that minimizes variations inducing possible category flips. When the classifier C_s is applied on the images produced by G_{ts}, only the samples annotated with the highest confidence are kept as pseudo-labeled samples. Hence, the probability of a wrong pseudo-label is negligible and even if a sample is mis-labeled in this phase, its impact on the final performance is not significant, as shown by of our ablation study.

4. Experimental Settings

MNIST $\rightarrow$ MNIST-M: MNIST has $60k$ images for training. As [1] we divided it into $50k$ samples for actual training and $10k$ for validation. All the $60k$ images from the MNIST-M training set were considered as test set. A subset of $1k$ images and their labels were also used to validate the classifier combination weights at test time.

USPS $\rightarrow$ MNIST: USPS has $6,562$ training, 729 validation, and $2,007$ test images. All of them were resized to 28×28 pixels. The $60k$ training images of MNIST were considered as test set, with $1k$ samples and their labels also used for validation purposes.

MNIST $\rightarrow$ USPS: even in this case MNIST training images were divided into $50k$ samples for actual training and $10k$ for validation. We tested on the whole set of $9,298$ images of USPS. Out of them, $1k$ USPS images and their labels were also used for validation.

SVHN $\rightarrow$ MNIST: SVHN contains over $600k$ color images of which $73,257$ samples are used for training and $26,032$ for validation while the remaining data are somewhat less difficult samples. We disregarded this last set and considered only the first two. The $60k$ MNIST training samples were considered as test set, with $1k$ MNIST images and their labels also used for validation.

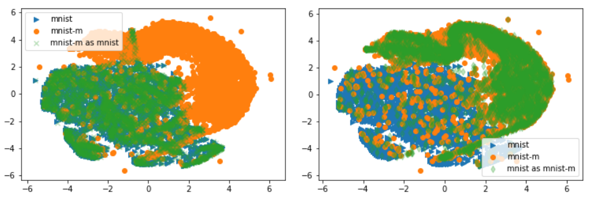
MNIST $\rightarrow$ SVHN: for MNIST we used again the $50k$/$10k$ training/validation sets. The whole set of $99,289$ SVHN samples was considered for testing with $1k$ images and their labels also used for validation.

Synth Signs $\rightarrow$ GTSRB: the Synth Signs dataset contains $100k$ images, out of which $90k$ were used for training and $10k$ for validation. The model was tested on the whole GTSRB dataset containing $51,839$ samples resized with bilinear interpolation to match the Synth Signs images' size of 40×40 pixels. Similarly to the previous cases, $1k$ GTSRB images and their labels were considered for validation purposes.

5. Distribution Visualizations

To visualize the original data distributions and their respective transformations we used t-SNE [5]. The images were pre-processed by scaling in $[-1, 1]$ and we applied PCA for dimensionality reduction from vectors with Width$\times$Height elements to 64 elements. Finally t-SNE with default parameters was applied to project data to a 2-dimensional space.

The behavior shown by the t-SNE data visualization presented in the main paper extends also for the other experimental settings. We integrate here the visualization for the MNIST$\rightarrow$MNIST-M case in Figure 1. The plots show again a successful mapping with the generated data that cover faithfully the target space.



(a) MNIST-M to MNIST (b) MNIST to MNIST-M

Figure 1: t-SNE visualization of source, target and source mapped to target images. Note how the mapped source covers faithfully the target space in all the settings.

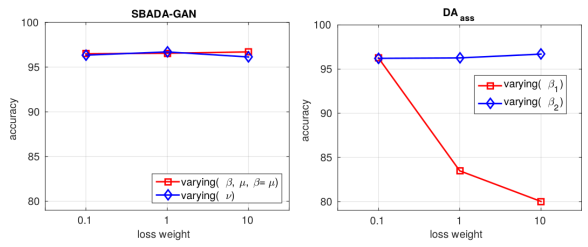


Figure 2: Behaviour of the SBADA-GAN and DA$_{ass}$ methods when changing their loss weights. (left) for SBADA-GAN we kept $\alpha = \gamma = 1$ and $\eta = 1$, while we varied alternatively the weights of the classification losses β, μ with $\beta = \mu$ and keeping $\nu = 1$, or the weight of the class consistency loss ν while fixing $\beta = \mu = 10$. (right) for the DA$_{ass}$ method we changed the weight of the walker loss β_1 while keeping that of the visit loss $\beta_2 = 0.1$, or alternatively we changed the weight of the visit loss β_2 while fixing that of the walker loss $\beta_1 = 1$.

6. Robustness experiments

The experiments about SBADA-GAN robustness to hyperparameters values are described at high level in Section 4.5 of the main paper submission. Here we report on the detailed results obtained on Synth. Signs $\rightarrow$ GTSRB when using SBADA-GAN and the DA$_{ass}$ method [3].

For SBADA-GAN we keep fixed the weights of the discriminative losses $\alpha = \gamma = 1$ as well as that of self-labeling $\eta = 1$, while we varied alternatively the weights of the classification losses β, μ or the weight of the class consistency loss ν in $[0.1, 1, 10]$. The results plotted in Figure 2 (left) show that the classification accuracy changes less than 0.2 percentage point. Furthermore, we used a batch size of 32 for our experiments and when reducing it to 16 the overall accuracy remains almost unchanged (from 96.7 to 96.5).

DA$_{ass}$ proposes to minimize the difference between the source and target by maximizing the associative similarity across domains. This is based on the two-step round-trip probability of an imaginary random walker starting from a sample (x_i^s, y_i) of the source domain, passing through an unlabeled sample of the target domain (x_j^t) and and returning to another source sample $(x_k^s, y_k = y_i)$ belonging to the same class of the initial one. This is formalized by first assuming that all the categories have equal probability both in source and in target, and then measuring the difference between the uniform distribution and the two-step probability through the so called *walker loss*. To avoid that only few target samples are visited multiple times, a second *visit loss* measures the difference between the uniform distribution and the probability of visiting some target samples. We tested the robustness of DA$_{ass}$ by using the code provided by its authors and changing the loss weights β_1 for the walker loss and β_2 for the visit loss in the same range used for the SBADA-GAN: [0.1 1 10]. Figure 2 (right) shows that DA$_{ass}$ is particularly sensitive to modifications of the visit loss weights which can cause a drop in performance of more than 16 percentage points. Moreover, the model assumption about the class balance sounds too strict for realistic scenarios: in practice DA$_{ass}$ needs every observed data batch to contain an equal number of samples from each category and reducing the number of samples from 24 to 12 per category causes a drop in performance of more than 4 percentage points from 96.3 to 92.8.

To conclude, although GAN methods are generally considered unstable and difficult to train, SBADA-GAN results much more robust than a not-GAN approach like DA$_{ass}$ to the loss weights hyperparameters and can be trained with small random batches of data while not losing its high accuracy performance.

References

[1] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan. Unsupervised pixel-level domain adaptation with gans. In *Computer Vision and Pattern Recognition (CVPR)*, 2017. 1, 2

[2] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky. Domain-adversarial training of neural networks. *J. Mach. Learn. Res.*, 17(1):2096–2030, 2016. 1

[3] P. Haeusser, T. Frerix, A. Mordvintsev, and D. Cremers. Associative domain adaptation. In *International Conference on Computer Vision (ICCV)*, 2017. 2

[4] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015. 1

[5] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008. 2