Unsupervised Domain Adaptation through CycleGAN

Pietro Basci
Politecnico di Torino
S266004@studenti.polito.it

Abstract

The objective of this work is to propose an approach to address the problem of domain adaptation through the use of generative adversarial networks. Specifically, it will be evaluated the effectiveness of CycleGAN in performing the image translation from the source to the target domain, and how much this operation can impact on the final image classification task in terms of accuracy.

CycleGAN is a GAN-based model which allows to perform unpaired image-to-image translation. It combines two GANs in a cyclical manner: the first generator allows to pass from the domain A to the domain B, then the second generator allows to come back from B to A.

The idea is to train a CycleGAN to translate images among source and target domains, extract the generator which allows to pass from source to target and use it either chained with the final CNN to produce "target-like" images on-the-fly as the training proceeds, or autonomously to generate offline a new version of the dataset which will be used to train the final CNN.

1. Introduction

Since their introduction, Convolutional Neural Networks allowed to achieve impressive results in performing image classification tasks. Usually to properly work, CNNs need to be trained on a very large amount of labeled data. However, gathering and labelling new images can be a very expensive and time-consuming task, so recently new approaches based on synthetic images has been considered to generate new labeled images.

When a CNN is trained, it is assumed that training and test data come from the same i.i.d. distribution. But most of the times, real life data and synthetic data belong to different distributions therefore performance can drop significantly. However, the problem is not limited to this case: visual appearance can also change due to seasonal and time changes, illumination conditions or simply the use of different sensors can also degrade the performance of visual recognition systems.

The Unsupervised Domain Adaptation aims to solve this issue through the use of unlabeled target data to mitigate the shift between source and target distributions. Usually, to address this problem two main approaches, both aiming at reducing the gap between the two domains, are followed: use generative models to produce new versions of target images so that they looks similar to source images (or vice versa) as in [7][8], or try to obtain a common representation of images from different domains in the latent space, i.e. obtaining domain-invariant features, as in [3][6].

In this work, the idea is to build a generative model able to "translate" images from the source domain to the target domain, train a Convolutional Neural Network on this new *target-like* version of the source images and evaluate the model performance on target images to state if this method can lead to any improvement.

As first step, a generative model will be trained to produce a new version of source images with the same "style" of the target images. To do that the CycleGAN proposed in [10] will be used. Since the "style" of interest is known and given that in a real application usually we do not have access to paired images as needed by Pix2Pix [5], the adversarial approach proposed with CycleGAN could be an adequate solution.

After that, a CNN will be trained on the new version of the source images. Specifically, the generative part obtained in the previous step, which allows to translate images from source to target domain, will be chained with the CNN so that it will "translate" images as the training proceeds or alternatively will be used to generate a new "target-like" version of the dataset to exploit for the CNN training. Finally, the CNN will be tested on target images to quantify the accuracy improvement.

2. Related Work

The problem of the Unsupervised Domain Adaptation has been already addressed in different works and it is still an active research area.

Ganin *et al.* [3] introduced a new architecture, that they call Domain-Adversarial Neural Network (DANN), which aims at obtaining a common representation of images from different domains in the latent space. The architecture includes a deep feature extractor and a deep label predictor.

To this standard feed-forward architecture, they added a domain classifier connected to the same feature extractor via a gradient reversal layer that multiplies the gradient by a negative constant during the backpropagation. Gradient reversal ensures that the feature distributions over the two domains are made similar, thus resulting in the domain-invariant features.

Sankaranarayanan *et al.* [6] proposed a new approach which, similarly to the previous case, tries to address the domain shift directly in the feature space by learning a joint feature space in which the distance between source and target distributions is minimized. Unlike the previous work, they used as adversarial branch an Auxiliary Classifier GAN (ACGAN) framework. Feature extractor and the ACGAN are updated so that both source and target embeddings produce source-like images.

Bousmalis *et al.* [7] proposed a GAN-based approach that adapts source images to appear as if drawn from the target domain; the classifier trained on such data outperformed several domain adaptation methods by large margins. Another similar approach was followed in [8].

Russo *et al.* [9] introduced a symmetric mapping among domains. They jointly optimize bi-directional image transformations combining them with target self-labeling. Besides the source-to-target transformation, they also considered the inverse target-to-source direction by using a symmetric architecture. The whole architecture is composed by: two generative models, which transform the source images to the target domain and vice versa; two discriminators, which discriminate real from generated images of source and target; and two classifiers which are trained to recognize respectively the original source images and their target-like transformed versions. At test time, both classifiers are applied respectively on the target images and on the transformed source-like target images. Their outputs are then linearly combined for the final prediction.

3. Dataset

To properly evaluate the performance of the model, four different benchmark datasets, containing digits belonging to 10 classes (0-9), with increasing difficulty will be considered: MNIST, USPS, MNIST-M, SVHN. Figure 1 shows some samples extracted from those datasets.

For being able later to evaluate the final CNN, some samples have been kept for the validation and test phases. Each dataset has been randomly split into three sets: training, validation and test set.

3.1. Overview

MNIST [1] is a huge dataset of handwritten digits containing 70k gray scale 28x28 pixel images of single digit on a black background. In the following experiments it will be used as source domain. The original dataset has been split into three sets: training set (50k images), validation set (10k) and test set (10k).

Figure 1: Examples of images in the considered datasets. Starting from the top: MNIST, USPS, MNIST-M, SVHN.

USPS [2] is another dataset of handwritten digits automatically scanned from envelopes by the U.S. Postal Service. It contains more than 9k gray scale 16x16 pixel images of single digit on a black background. It will be used as a target domain. Also in this case, the dataset has been split in three sets: training (5k images), validation (2k) and test (2k).

MNIST-M [3] is a variant of the MNIST dataset where the background is substituted by a random crop uniformly sampled from BSDS500. Therefore, it contains the same 70k images of handwritten digits, but this time they are RGB 28x28 pixel images of single digit on a colored background. It will be used as a target domain. The original dataset has been split following the same strategy used for MNIST.

SVHN [4] is the most challenging dataset considered. It was obtained by cropping house number in Google Street View images and so it contains much more diversity: beside the variety of style, images include some additional digits near to the centered one. It is composed by about 100k RGB 32x32 pixel images. Also this dataset will be used as a target domain. The dataset has been split in three sets: training (63k images), validation (10k) and test (26k).

3.2. Preprocessing

Images of each dataset have been transformed so that they have the same characteristics: all images have been resized to 32x32 pixel and converted into RGB, unlike some previous works which consider an easier scenario with only grayscale images.

Moreover, images have been scaled so that the value of each pixel's channel falls in the range [-1, 1]. Given the large number of images in the considered datasets, no data augmentation techniques have been applied.

4. Methods and algorithms

Unsupervised Domain Adaptation aims at exploit knowledge acquired from the source domain, where a large number of annotated images are available, on the target domain where there are no labeled images.

Assuming that there is access to a labeled dataset drawn from the source domain and to a fully unlabeled dataset drawn from the target domain. The objective is to maximize the accuracy of the final classifier on target images, while training only on source. When the gap between the distributions of the two domains is too large, the knowledge learned by the model on the source generally leads to a very poor results on target images. The effect of the domain shift on the features extracted can be seen in Figure 2.

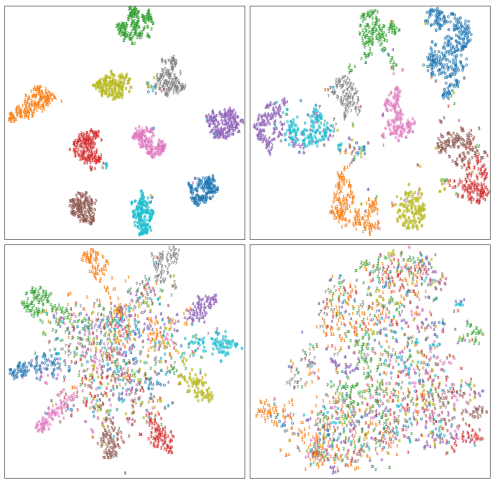

Figure 2: t-SNE visualization of features extracted by the last convolutional layer of the CNN trained on MNIST. *Top left*: feature extracted on MNIST. *Top right*: features extracted on USPS. *Bottom left*: features extracted on MNIST-M. *Bottom right*: feature extracted on SVHN.

In order to reduce this gap and try to improve the final model's performance, a generative model has been used to transform the source images such that they appear similar to the target images. This approach allows to disengage the process of domain adaptation from the specific task, therefore it can be easily combined with any model without any changes.

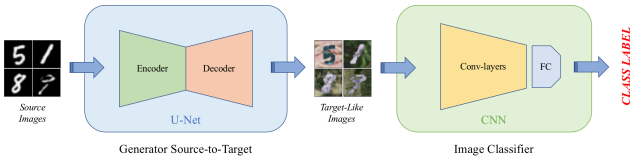

Figure 3: The proposed method. On the left the generator used to translate images from Source to Target; on the right a standard CNN trained on the new version of images.

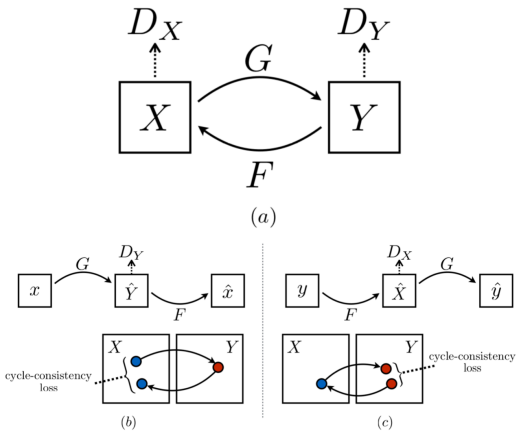

Figure 4: (a) CycleGAN architecture: two generators G and F which allows to pass from one domain to the other and two discriminators D_X and D_Y which distinguish between *real/fake* images. (b) Forward cycle-consistency loss. (c) Backward cycle-consistency loss.

Usually, source and target datasets contain unpaired images, therefore CycleGAN [10] has been considered as a suitable approach to solve this specific task. Once the model has been trained to transform images, the generator which allows to pass from source to target has been used to adapt the images used for the training. Specifically, this module can be used either chained with the final CNN to produce "target-like" images on-the-fly as the training proceeds, or autonomously to produce offline a new version of the dataset which will be used to train the final CNN.

The choice between the two approaches could be driven by hardware constraints: the first method will require no more disk space, but it can slow down the training process as it must apply transformations to images; the second will not affect the training process, but it will require much more space to store the new version of the dataset.

4.1. CycleGAN

CycleGAN [10] is a generative model which allows to perform image-to-image translation. The main difference with respect to other works, such as Pix2Pix [5], is that it does not require paired imaged to learn the mapping functions among the two domains. This aspect makes the system suitable in most of the real case where no paired images are available.

The model, showed in Figure 4, contains two mapping functions $G: X \rightarrow Y$ and $F: Y \rightarrow X$, and two associated adversarial discriminators D_Y and D_X. The first encourage G to translate samples from X into images indistinguishable from domain Y, the second does the same for F on images from Y to X. To further regularize the mappings, they introduced two *cycle consistency losses* that capture the intuition that translating from one domain to the other and back again should lead at the starting point: forward cycle-consistency loss and backward cycle-consistency loss.

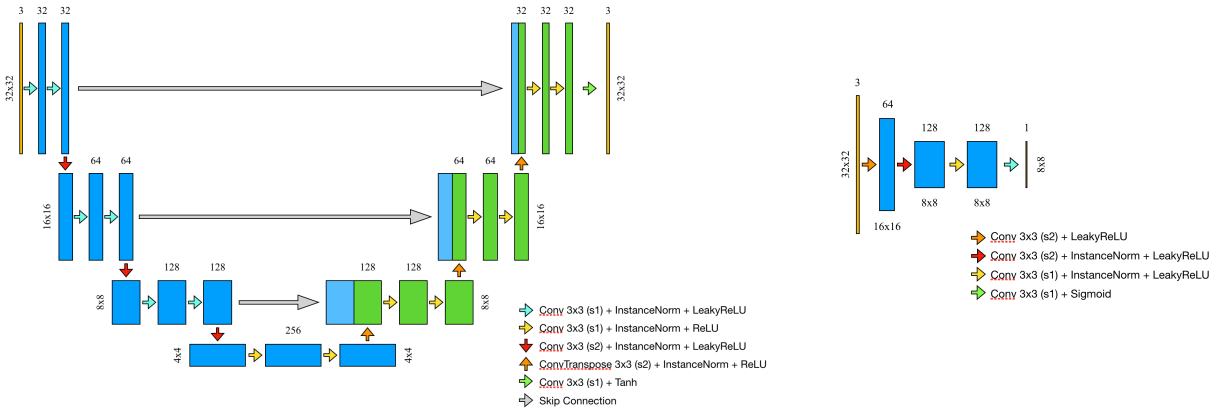

Figure 5: Architectures used in this work. *Left*: U-Net used as generator. *Right*: PatchGAN used as discriminator.

4.1.1 Generator

The architecture of the generator used in this work is derived from U-Net proposed by Ronneberger *et al.* [11] for biomedical image segmentation. U-Net is an encoder-decoder model with skip connections between layers with the same sized feature maps which allow some information to circumvent the bottleneck. U-Net has been reused also for image translation task in Pix2Pix [5].

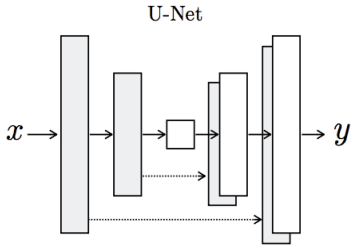

Figure 6: U-Net architecture.

The architecture considered uses only blocks of Conv2D-InstanceNorm-ReLU with 3x3 filters and stride 2 during the downsampling and TransposeConv2D-InstanceNorm-LeakyReLU during the upsampling. Moreover, each downsampling/upsampling block is followed by 2 other blocks with stride 1. BatchNormalization has been replaced with InstanceNormalization [13] which standardizes the values on each feature map and allows to improve performance in image generation task. According to [14], all network weights have been initialized from a Gaussian distribution with a 0 mean and a standard deviation of 0.02. The final model is shown in Figure 5.

4.1.2 Discriminator

The discriminator used is the PatchGAN proposed in [12]. This model tries to classify as real/fake each *NxN* patch of the input image. It is a fully convolutional architecture which produces in output a matrix where each value is the final prediction for a patch of the image.

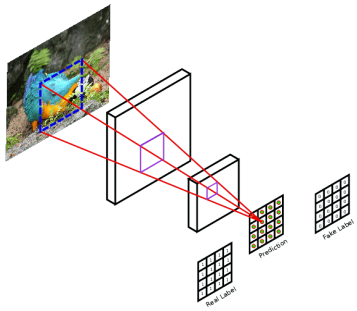

Figure 7: PatchGAN architecture.

The architecture has been tailored for the specific case and so that each value in the final prediction refers to a 23x23 patch on the original image. It uses only blocks of Conv2D-InstanceNorm-LeakyReLU with 3x3 filters and stride 2. Also in this case, Instance Normalization has been used and all network weights have been initialized from a Gaussian distribution with a 0 mean and a standard deviation of 0.02. The final model is shown in Figure 5.

4.1.3 Loss function

In CycleGAN, the problem can be formulated as a minimax game where the generators learn to fool the discriminators and the discriminators learn to find fake images. The training proceeds trying to solve:

$$arg\min_{G,F}\max_{D_X,D_Y} \mathcal{L}(G, F, D_X, D_Y)$$

The full objective is defined as:

$$\mathcal{L}(G, F, D_X, D_Y) = \mathcal{L}_{GAN}(G, D_Y, X, Y) + \mathcal{L}_{GAN}(F, D_X, Y, X) + \lambda_1 \mathcal{L}_{cyc}(G, F) + \lambda_2 \mathcal{L}_{identity}(G, F)$$

For the *adversarial loss* according to [10], the negative log likelihood has been replaced by a least-square loss. This loss showed to be more stable during training and generates better results. It helps against mode collapse and vanishing gradients. In particular, the generator and the discriminator try to minimize respectively:

$$\mathbb{E}_{x \sim p_{data}(x)}[(D(G(x)) - 1)^2]$$

$$\mathbb{E}_{y \sim p_{data}(y)}[(D(y) - 1)^2] + \mathbb{E}_{x \sim p_{data}(x)}[D(G(x))^2]$$

The *cycle consistency loss* is used to guarantee that after translating from one domain to the other, translating back again should lead to the original image. It is defined as:

$$\mathcal{L}_{cyc}(G, F) = \mathbb{E}_{x \sim p_{data}(x)}[\|F(G(x)) - x\|_1] + \mathbb{E}_{y \sim p_{data}(y)}[\|G(F(y)) - y\|_1]$$

Finally, the *identity loss* is used to help color preservation between input and output and is defined as:

$$\mathcal{L}_{identity}(G, F) = \mathbb{E}_{y \sim p_{data}(y)}[\|(G(y) - y\|_1] + \mathbb{E}_{x \sim p_{data}(x)}[\|F(x) - x\|_1]$$

4.2. CNN

For the image classification task, a standard Convolutional Neural Network has been considered. For all the experiments, the CNN topology used for the classification was equal to the one used in [6] and similar to ones used in other papers, in order to be comparable with previous work in unsupervised domain adaptation. The architecture is described in Figure 8.

Conv 5x5 (64) ReLU MaxPool 2x2	Conv 5x5 (64) ReLU MaxPool 2x2	Conv 5x5 (64) ReLU	FC 128 units ReLU	FC 10 units SoftMax

Figure 8: CNN architecture.

4.3. Training

The training proceeds in two steps: firstly, the CycleGAN model must be trained, then it is possible to train the CNN on the new version of the images. The most challenging part is the training of the first model.

All networks were trained from scratch. For all experiments, in the CycleGAN objective function, λ_1 was set to 8 and λ_2 was set to $0.1 \cdot \lambda_1$. The optimizer used was Adam with a learning rate of 0.0002. The batch size considered was 32. The model was trained for 50 epochs. During the training, in order to slow down the learning process of the discriminators with respect to the generators, the objective of the discriminators have been divided by 2.

Experiments showed that the learning process leads the generator to produce better results when the discriminator accuracy varies in the range 60-80%. The trends of the generators and discriminators losses and accuracies were quite similar for all the couples of datasets analysed and are showed in Figure 9.

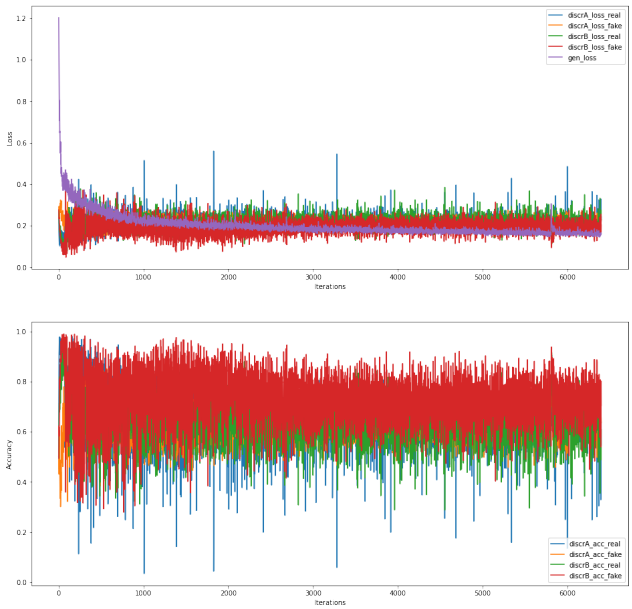

Figure 9: CycleGAN Losses and Accuracies trends.

As in [5], at inference time the generators were used as during the training phase: the use of the dropout also during inference is exploited to introduce randomness in the image generation process. This method should ensure that the model produce different images at each iteration also using the same input. However, experiments showed that it introduces only limited changes on the output image.

About the training of the CNN, also in this case the network was trained from scratch, using the Adam solver with a learning rate of 0.005.

5. Evaluation

The method proposed has been evaluated on three different couples of datasets with increasing domain shift, considering always MNIST as the source domain. Outcomes are expressed in terms of qualitative and

Method	MNIST → MNIST-M	MNIST → USPS	MNIST → SVHN
Source Only	60.4 ± 0.5	79.3 ± 3.5	28.73 ± 1.8
DANN [3]	77.4	85.1	35.7
CoGAN [15]	62.0	91.2	-
ADDA [16]	-	89.4 ± 0.2	-
PiexlDA [7]	98.2	95.9	-
GenToAdapt [6]	-	92.5 ± 0.7	36.4 ± 1.2
SBADAGAN [9]	99.4	97.6	61.1
Target Only	96.9 ± 0.2	95.6 ± 0.4	84.6 ± 3.4
CycleGAN Adaptation	83.7 ± 0.4	85.4 ± 1.6	37.0 ± 0.9

Table 1: Results comparison against other works in terms of accuracy. *Source Only* and *Target Only* refer to accuracies achieved on the target domain without domain adaptation, by using for the training only the source domain and the target domain respectively.

quantitative results: the firsts highlight the ability of the generator to perform the domain adaptation from source to target; the seconds shows the improvements introduced by this method in terms of classification accuracy and the comparison with the main related works.

5.1. Quantitative Results

Table 1 shows a comparison of results against other works. Results are expressed in terms of accuracy achieved by the CNN after the domain adaptation. It is possible to see that CycleGAN leads to an improvement of about 24% on the couple MNIST-to-MNIST-M which is comparable or even better that other works [3][15]. A more limited improvement (about 6%) but still comparable with [3] is achieved on MNIST-to-USPS adaptation, despite the similarity among the two datasets. Instead, the worst results are obtained on the couple MNIST-to-SVHN where the accuracy still remains quite low, with an increase of only 9%. However, also many of the approaches proposed in previous works suffer on this couple of domains, due to the high style difference among images.

5.2. Qualitative Results

Figure 10 shows some images generated by the model compared to similar real images. The best results are obtained in the translation MNIST-to-MNIST-M where fake images are very similar to real ones. Those results are due to the fact that images of the two datasets have the same shape and the only difference is given by the background color. The worst results are achieved in the translation MNIST-to-SVHN where the domain shift is quite large. SVHN contains images which are different in terms of font, size, rotation and colors. Moreover, they also contain some additional digits to the side of the main centered one.

Therefore, the proposed method shows to work better when the differences between the domains are quite limited, for instance due to color, illumination and noise, rather than domains with differences due to geometric variations. This aspect makes the system suitable to adapt synthetic datasets so that they appear more real.

Figure 11 shows the effect of the adaptation on the domains. Before the adaptation the clouds of points are clearly separated, while after the adaptation the new version

Figure 10: Images produced by the generator. *First row*: input images (MNIST). *Following couples of rows*: generated images (first row) and the most similar real images found with the Nearest Neighbor algorithm (second row) for the MNIST-M, USPS and SVHN domain, respectively.

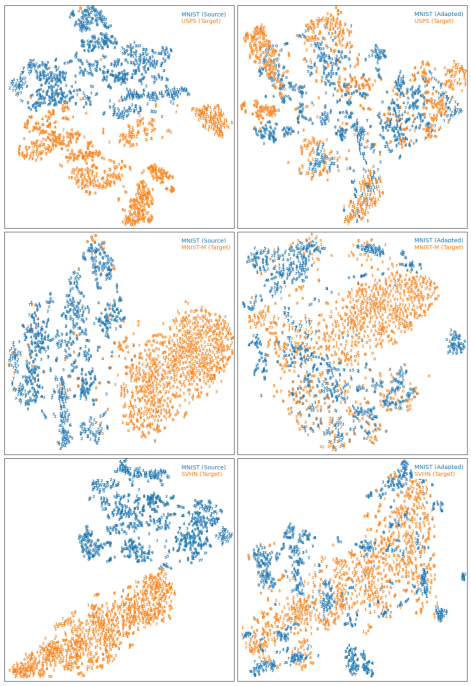

Figure 11: t-SNE visualization of Source domain, before (left column) and after (right column) adaptation, and Target domain. *First row*: MNIST-to-USPS. *Second row*: MNIST-to-MNIST-M. *Third row*: MNIST-to-SVHN.

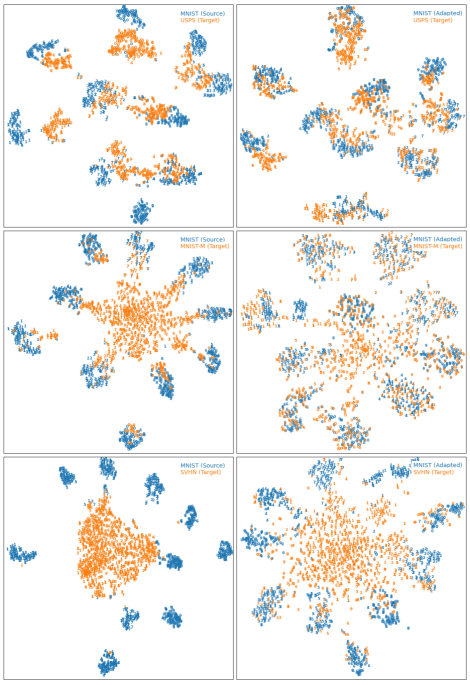

Figure 12: t-SNE visualization of features extracted on Source and Target samples, by the last convolutional layer of the CNN, trained on Source (left column) and on Source adapted (right column). *First row*: MNIST-to-USPS. *Second row*: MNIST-to-MNIST-M. *Third row*: MNIST-to-SVHN.

of the dataset completely overlap with the target dataset. This means that the generator reduced the gap among the two distributions by transferring the target style to the source images. Figure 12 instead shows the effect of the adaptation on the features extracted by the last convolutional layer of the CNN trained on the original dataset and on the adapted dataset. Finally, the improvement introduced by the adaptation can also be seen on confusion matrices showed in Figure 13.

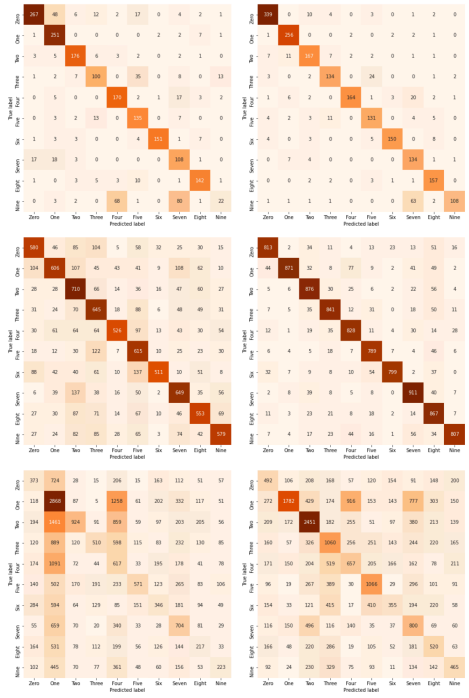

Figure 13: Confusion Matrices computed on the test set using the CNN trained on MNIST before (left column) and after (right column) adaptation. *First row*: MNIST-to-USPS. *Second row*: MNIST-to-MNIST-M. *Third row*: MNIST-to-SVHN.

6. Conclusion

In this work the problem of domain adaptation has been addressed. The analysis proposed showed that CycleGAN is a valid method to mitigate issues related to domain shift. The first generator inside CycleGAN, allows to reduce the gap between data distributions, thus leading to a significant improvement of the classifier performance, without the use of target labels. Using a CNN trained on the new target-like version of the dataset, allows to improve its accuracy on the real target images.

Experiments highlighted that these improvements were more notable when the differences between the domains are not very large. Therefore, this can be considered a valid approach to fill the gap between synthetic and real images, making synthetic images more realistic. Due to hardware

and time constraints, the results reported in this paper were obtained considering a CycleGAN training for only 50 epochs. Despite that, the method proposed achieved significant results. Given the losses trends, it seems likely that a longer training can induce the generator to produce higher quality images, thus leading the classifier to a further performance improvement.

7. Future work

As a future work, firstly, it is possible to train the model using a higher number of epochs to further improve the ability of the generator to produce images similar to real ones. Then, since CycleGAN includes two generators to pass from one domain to the other and vice versa, it is possible to exploit both. Specifically, two classifiers can be used at test time to produce the final label: one trained on the transformed dataset and the other trained on the original source dataset. Hence, the first generator is used to generate a target-like version of the dataset to train the first classifier and the second is used at test time to produce on-the-fly a source-like version of the image before the classification. The final prediction will be a linear combination of the output of the two classifiers. A similar approach was also followed in [9].

References

[1] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11):2278–2324, 1998.

[2] J. Friedman, T. Hastie, and R. Tibshirani. The elements of statistical learning, volume 1. Springer series in statistics Springer, Berlin, 2001.

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

[4] Y.Netzer, T.Wang,A.Coates,A.Bissacco,B.Wu,and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. In NIPS workshop on deep learning and unsupervised feature learning, volume 2011, page 5, 2011.

[5] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image- to-image translation with conditional adversarial networks. In CVPR, 2017.

[6] S. Sankaranarayanan, Y. Balaji, C. D. Castillo, and R. Chellappa. Generate to adapt: Aligning domains using generative adversarial networks. arXiv preprint arXiv:1704.01705, 2017.

[7] 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.

[8] Y. Taigman, A. Polyak, and L. Wolf. Unsupervised cross-domain image generation. In International Conference on Learning Representations (ICLR), 2017.

[9] P. Russo, F. M. Carlucci, T. Tommasi, and B. Caputo. From source to target and back: symmetric bi-directional adaptive gan. In Computer Vision and Pattern Recognition (CVPR), 2018.

[10] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adver- sarial networks. arXiv preprint arXiv:1703.10593, 2017.

[11] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In MIC-CAI, 2015.

[12] C. Li and M. Wand. Precomputed real-time texture synthesis with markovian generative adversarial networks. ECCV, 2016.

[13] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Instance normalization: The missing ingredient for fast styliza- tion. arXiv preprint arXiv 1607.08022, 2016.

[14] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional gen- erative adversarial networks. In ICLR, 2016.

[15] M.-Y. Liu and O. Tuzel. Coupled generative adversarial networks. In Neural Information Processing Systems (NIPS). 2016.

[16] E. Tzeng, J. Hoffman, T. Darrell, and K. Saenko. Adversarial discriminative domain adaptation. In Computer Vision and Pattern Recognition (CVPR), 2017.