



TriGAN: image-to-image translation for multi-source domain adaptation

Subhankar Roy¹ · Aliaksandr Siarohin¹ · Enver Sangineto¹ · Nicu Sebe¹ · Elisa Ricci^{1,2}

Received: 24 June 2020 / Revised: 14 September 2020 / Accepted: 11 December 2020 / Published online: 19 January 2021
© The Author(s) 2021

Abstract

Most domain adaptation methods consider the problem of transferring knowledge to the target domain from a single-source dataset. However, in practical applications, we typically have access to multiple sources. In this paper we propose the first approach for multi-source domain adaptation (MSDA) based on generative adversarial networks. Our method is inspired by the observation that the appearance of a given image depends on three factors: the *domain*, the *style* (characterized in terms of low-level features variations) and the *content*. For this reason, we propose to project the source image features onto a space where only the dependence from the content is kept, and then re-project this invariant representation onto the pixel space using the target domain and style. In this way, new labeled images can be generated which are used to train a final target classifier. We test our approach using common MSDA benchmarks, showing that it outperforms state-of-the-art methods.

Keywords Unsupervised domain adaptation · Generative adversarial network · Image classification · Image-to-image translation

1 Introduction

A well-known problem in computer vision is the need to adapt a classifier trained on a given *source* domain in order to work on a different *target* domain. Since the two domains typically have different marginal feature distributions, the adaptation process needs to reduce the corresponding *domain shift* [44]. In many practical scenarios, the target data are not annotated and unsupervised domain adaptation (UDA) methods are required.

While most previous adaptation approaches consider a single-source domain, in real-world applications we may have access to multiple datasets. In this case, multi-source domain adaptation (MSDA) methods [29,34,50,51] may be

adopted, in which more than one source dataset is considered in order to make the adaptation process more robust. However, although more data can be used, MSDA is challenging as multiple domain-shift problems need to be simultaneously and coherently solved.

In this paper we deal with (unsupervised) MSDA using a data-augmentation approach based on a generative adversarial network (GAN) [12]. Specifically, we generate artificial target samples by “translating” images from all the source domains into target-like images. Then, the synthetically generated images are used for training the target classifier. While this strategy has been recently adopted in the single-source UDA scenario [16,25,32,39,40], we are the first to show how it can be effectively used in a MSDA setting. In more detail, our goal is to build and train a “universal” translator which can transform an image from an input domain to a target domain. The translator network is “universal” because it is not specific for a given source dataset but can transform images from multiple source domains into the target domain, given a domain label as input (Fig. 1). The proposed translator is based on an *encoder*, which extracts domain-invariant intermediate features, and a *decoder*, which projects these features onto the domain-specific target distribution.

To make this image translation effective, we assume that the appearance of an image depends on three factors: the *con-*

Supplementary Information The online version contains supplementary material available at <https://doi.org/10.1007/s00138-020-01164-4>.

✉ Enver Sangineto
enver.sangineto@unitn.it
Subhankar Roy
subhankar.roy@unitn.it

¹ Department of Information Engineering and Computer Science (DISI), University of Trento, Trento, Italy

² TeV, Fondazione Bruno Kessler (FBK), 38050 Trento, Italy

content, the *domain* and the *style*. The *domain* models properties that are shared by the elements of a dataset but which may not be shared by other datasets. On the other hand, the *style* factor represents properties which are shared among different *local* parts of a *single image* and describes low-level features which concern a specific image (e.g., the color or the texture). The *content* is the semantics that we want to keep unchanged during the translation process: typically, it is the foreground object shape which corresponds to the image label associated with each source data sample. Our encoder obtains the intermediate representations in a two-step process: we first generate style-invariant representations and then we compute the domain-invariant representations. Symmetrically, the decoder transforms the intermediate representations, first projecting these features onto a domain-specific distribution and then onto a style-specific distribution. In order to modify the underlying distribution of a set of features, inspired by [37], in the encoder we use *whitening* layers which progressively align the style-and-domain feature distributions. Then, in the decoder, we project the intermediate invariant representation onto a new domain- and style-specific distribution with *Whitening and Coloring (WC)* [41] batch transformations, according to the target data.

A “universal” translator similar in spirit to our proposed generator is StarGAN [4]. The goal of StarGAN is pure image translation and it is not used for discriminative tasks (e.g., UDA tasks). The main advantage of a universal translator with respect to train N -specific source-to-target translators (being N the number of source domains) is that the former can jointly use all the source datasets, thus alleviating the risk of overfitting [4]. However, differently from our proposed generator, in StarGAN the domain label is represented by a one-hot vector concatenated with the input image. As shown in [41] this procedure is less effective than using domain-label conditioned batch transforms. We empirically show that, when we use StarGAN in our MSDA scenario, the synthesized images are much less effective for training the target classifier, which confirms that our batch-based transformations of the image distribution are more effective for our translation task.

Contributions Our main contributions can be summarized as follows. (i) We propose the first generative MSDA method. We call our approach TriGAN because it is based on representing the image appearance using three different factors: the style, the domain and the content. (ii) The proposed image translation process is based on style- and domain-specific statistics which are first removed from and then added to the source images by means of modified *WC* layers. Specifically, we use the following feature transformations (associated with a corresponding layer type): Instance Whitening Transform (*IWT*), Domain Whitening Transform (*DWT*) [37], conditional Domain Whitening Transform (*cDWT*) and Adaptive Instance Whitening Transform (*AdaIWT*). *IWT*

and *AdaIWT* are novel layers introduced in this paper. (iii) We test our method on two MSDA datasets: Digits-Five [50] and Office-Caltech10 [11], outperforming state-of-the-art methods.

2 Related work

In this section we review the previous approaches on UDA, considering both single-source and multi-source methods. Since the proposed generator is also related to deep models used for image-to-image translation, we also analyze related work on this topic.

Single-source UDA Single-source UDA approaches assume a single labeled source domain and can be broadly classified under three main categories, depending on the strategy adopted to cope with the domain-shift problem. The first category uses first- and second-order statistics to model the source and the target feature distributions. For instance, [26, 27,47,49] minimize the Maximum Mean Discrepancy, i.e., the distance between the mean of feature distributions between the two domains. On the other hand, [31,35,43] achieve domain invariance by aligning the second-order statistics through correlation alignment. Differently, [2,24, 28] reduce the domain shift by domain alignment layers derived from batch normalization (BN) [19]. This idea has been recently extended in [37], where grouped-feature whitening (DWT) is used instead of feature standardization as in *BN*. In our proposed encoder we also use the DWT layers, which we adapt to work in a generative network. In addition, we also propose other style- and domain-dependent batch-based normalizations (i.e., *IWT*, *cDWT* and *AdaIWT*).

The second category of methods computes domain-agnostic representations by means of an adversarial learning-based approach. For instance, discriminative domain-invariant representations are constructed through a gradient reversal layer in [8]. Similarly, Tzeng et al. [45] use a domain confusion loss and a domain discriminator to align the source and the target domain.

The third category of methods uses adversarial learning in a generative framework (i.e., GANs [12]) to create artificial source and/or target images and perform domain adaptation. Notable approaches are SBADA-GAN [39], CyCADA [16], CoGAN [25], I2I Adapt [32] and Generate To Adapt (GTA) [40]. While these generative methods have been shown to be very successful in UDA, none of them deals with a multi-source setting. Note that trivially extending these approaches to an MSDA scenario involves training N different generators, being N the number of source domains. In contrast, in our universal translator, only a subset of parameters grow linearly with the number of domains (Sect. 3.2.3), while the others are shared over all the domains.

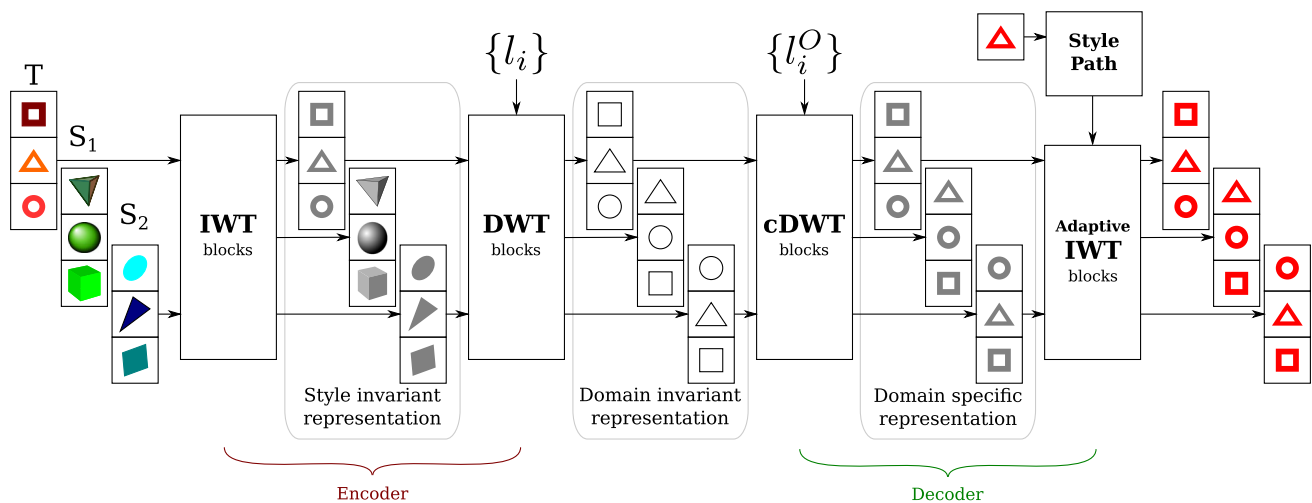


Fig. 1 An overview of the TriGAN generator. We schematically show three domains $\{T, S_1, S_2\}$ —objects with holes, 3D objects and skewed objects, respectively. The content is represented by the object’s shape—square, circle or triangle. The style is represented by the color: each image input to \mathcal{G} has a different color and each domain has its own set of styles. First, the encoder \mathcal{E} creates a style-invariant representation using IWT blocks. DWT blocks are then used to obtain a domain-invariant rep-

resentation. Symmetrically, the decoder \mathcal{D} brings back domain-specific information with cDWT blocks (for simplicity we show only a single output domain, T). Finally, we apply a reference style. The reference style is extracted using the style path and it is applied using the Adaptive IWT blocks. In this figure, l_i and l_i^o denote, respectively, the input and the output domain labels

Moreover, since we train our generator using $(N+1)^2$ translation directions, we can largely increase the number of training sample-domain pairs effectively used (Sect. 3.3).

Multi-source UDA In [51], multiple-source knowledge transfer is obtained by borrowing knowledge from the target k nearest-neighbor sources. Similarly, a distribution weighted combining rule is proposed in [29] to construct a target hypothesis as a weighted combination of source hypotheses. Recently, Deep Cocktail Network (DCTN) [50] uses the distribution-weighted combining rule in an adversarial setting. A Moment Matching Network (M³SDA) is introduced in [34] to reduce the discrepancy between the multiple-source and the target domains. Zhao et al. [52] investigate multi-source domain adaptation for segmentation tasks, while Rakshit et al. [36] adversarially train an ensemble of source domain classifiers in order to align the source domains to each other. Adversarial training is used also in [53], where the authors propose to use the Wasserstein distance between the source samples and the target distribution in order to select those samples which are the closest to the target domain.

Differently from these methods which operate in a discriminative setting, we propose the first generative approach for an MSDA scenario, where the target dataset is populated with artificial “translations” of the source images.

Image-to-image translation Image-to-image translation approaches, i.e., those methods which transform an image from one domain to another, possibly keeping its semantics, are the basis of our method. In [20] a U-Net network translates images under the assumption that paired images in

the two domains are available at training time. In contrast, CycleGAN [54] can learn to translate images using unpaired training samples. Note that, by design, these methods work with two domains. ComboGAN [1] partially alleviates this issue by using N generators for translations among N domains. Our work is also related to StarGAN [4] which handles unpaired image translation among N domains ($N \geq 2$) through a single generator. However, StarGAN achieves image translation without explicitly forcing the image representations to be domain invariant, and this may lead to a significant reduction of the network representation power as the number of domains increases. On the other hand, our goal is to obtain an explicit, intermediate image representation which is style-and-domain independent. We use IWT and DWT to achieve this. We also show that this invariant representation can simplify the re-projection process onto a desired style and target domain. This is achieved through AdaIWT and cDWT which results into very realistic translations amongst domains. Very recently, a whitening and coloring-based image-to-image translation method was proposed in [3], where the whitening operation is *weight-based*: the transformation is embedded into the network weights. Specifically, whitening is approximated by enforcing the covariance matrix, computed using the intermediate features, to be equal to the identity matrix. Conversely, our whitening transformation is *data dependent* (i.e., it depends on the specific batch statistics, Sect. 3.2.1) and uses the Cholesky decomposition [5] to compute the whitening matrices of the

input samples in a closed form, thereby eliminating the need of additional ad hoc losses.

3 Style-and-domain-based image translation

In this section we describe the proposed approach for MSDA. We first provide an overview of our method and we introduce the notation adopted throughout the paper (Sect. 3.1). Then, we describe the TriGAN architecture (Sect. 3.2) and our training procedure (Sect. 3.3).

3.1 Notation and overview

In the MSDA scenario we have access to N labeled source datasets $\{S_j\}_{j=1}^N$, where $S_j = \{(\mathbf{x}_k, y_k)\}_{k=1}^{n_j}$, and a target unlabeled dataset $T = \{\mathbf{x}_k\}_{k=1}^{n_t}$. All the datasets (target included) share the same semantic categories, and each of them is associated with a domain $\mathbf{D}_1^s, \dots, \mathbf{D}_N^s, \mathbf{D}^t$, respectively. Our final goal is to build a classifier for the target domain \mathbf{D}_t exploiting the data in $\{S_j\}_{j=1}^N \cup T$.

Our method is based on two separate training stages. We initially train a generator \mathcal{G} which learns how to change the appearance of a real input image in order to adhere to a desired domain and style. \mathcal{G} learns $(N + 1)^2$ mappings between every possible pair of image domains, in this way exploiting much more supervisory information with respect to a plain strategy in which N different source-to-target generators are separately trained [4] (Sect. 3.3). Once \mathcal{G} is trained, in the second stage we use it to generate target data having the same content of the source data, thus creating a new, *labeled*, target dataset, which is finally used to train a target classifier \mathcal{C} . However, in training \mathcal{G} (first stage), we do not use class labels and T is treated in the same way as the other datasets.

As mentioned in Sect. 1, \mathcal{G} is composed of an encoder \mathcal{E} and a decoder \mathcal{D} (Fig. 1). The role of \mathcal{E} is to “whiten,” i.e., to remove, both domain-specific and style-specific aspects of the input image features in order to obtain domain- and style-invariant representations. Symmetrically, \mathcal{D} “colors” the domain- and style-invariant features generated by \mathcal{E} , by progressively projecting these intermediate representations onto a domain- and style-specific space.

In the first training stage, \mathcal{G} takes as input a batch of images $B = \{\mathbf{x}_1, \dots, \mathbf{x}_m\}$ with corresponding *domain* labels $L = \{l_1, \dots, l_m\}$, where \mathbf{x}_i belongs to the domain \mathbf{D}_{l_i} and $l_i \in \{1, \dots, N + 1\}$. Moreover, \mathcal{G} takes as input a batch of output domain labels $L^O = \{l_1^O, \dots, l_m^O\}$, and a batch of reference style images $B^O = \{\mathbf{x}_1^O, \dots, \mathbf{x}_m^O\}$, such that \mathbf{x}_i^O has domain label l_i^O . For a given $\mathbf{x}_i \in B$, the task of \mathcal{G} is to transform \mathbf{x}_i into $\hat{\mathbf{x}}_i$ such that (1) \mathbf{x}_i and $\hat{\mathbf{x}}_i$ share the same content but (2) $\hat{\mathbf{x}}_i$ belongs to domain $\mathbf{D}_{l_i^O}$ and has the same style of \mathbf{x}_i^O .

3.2 TriGAN architecture

The TriGAN architecture is composed of a generator network \mathcal{G} and a discriminator network $\mathcal{D}_{\mathcal{P}}$. As above mentioned, \mathcal{G} comprises an encoder \mathcal{E} and decoder \mathcal{D} , which we describe in (Sects. 3.2.2–3.2.3). The discriminator $\mathcal{D}_{\mathcal{P}}$ is based on the Projection Discriminator [30]). Before describing the details of \mathcal{G} , we briefly review the *WC* transform [41]) (Sect. 3.2.1) which is used as the basic operation in our proposed batch-based feature transformations.

3.2.1 Preliminaries: whitening and coloring transform

Let $F(\mathbf{x}) \in \mathbb{R}^{h \times w \times d}$ be the tensor representing the activation values of the convolutional feature maps in a given layer corresponding to the input image \mathbf{x} , with d channels and $h \times w$ spatial locations. We treat each spatial location as a d -dimensional vector; thus, each image \mathbf{x}_i contains a set of vectors $X_i = \{\mathbf{v}_1, \dots, \mathbf{v}_{h \times w}\}$. With a slight abuse of the notation, we use $B = \cup_{i=1}^m X_i = \{\mathbf{v}_1, \dots, \mathbf{v}_{h \times w \times m}\}$, which includes all the spatial locations in all the images in a batch. The *WC* transform is a multivariate extension of the per-dimension normalization and shift-scaling transform (*BN*) proposed in [19] and widely adopted in both generative and discriminative networks. *WC* can be described by:

$$WC(\mathbf{v}_j; B, \boldsymbol{\beta}, \boldsymbol{\Gamma}) = \text{Coloring}(\bar{\mathbf{v}}_j; \boldsymbol{\beta}, \boldsymbol{\Gamma}) = \boldsymbol{\Gamma} \bar{\mathbf{v}}_j + \boldsymbol{\beta} \quad (1)$$

where:

$$\bar{\mathbf{v}}_j = \text{Whitening}(\mathbf{v}_j; B) = \mathbf{W}_B(\mathbf{v}_j - \boldsymbol{\mu}_B). \quad (2)$$

In Eq. 2, $\boldsymbol{\mu}_B$ is the centroid of the elements in B , while \mathbf{W}_B is such that $\mathbf{W}_B^T \mathbf{W}_B = \boldsymbol{\Sigma}_B^{-1}$, where $\boldsymbol{\Sigma}_B$ is the covariance matrix computed using B . The result of applying Eq. 2 to the elements of B , is a set of *whitened* features $\bar{B} = \{\bar{\mathbf{v}}_1, \dots, \bar{\mathbf{v}}_{h \times w \times m}\}$, which lie in a spherical distribution, (i.e., with a covariance matrix equal to the identity matrix). On the other hand, Eq. 1 performs a *coloring* transform, i.e., projects the elements in \bar{B} onto a learned multivariate Gaussian distribution. While $\boldsymbol{\mu}_B$ and \mathbf{W}_B are computed using the elements in B (they are data dependent), Eq. 1 depends on the d -dimensional learned parameter vector $\boldsymbol{\beta}$ and on the $d \times d$ -dimensional learned parameter matrix $\boldsymbol{\Gamma}$. Equation 1 is a linear operation and can be simply implemented using a convolutional layer with kernel size $1 \times 1 \times d$. We refer to [41] for more details on how *WC* can be efficiently implemented.

In this paper we use the *WC* transform in our encoder \mathcal{E} and decoder \mathcal{D} , in order to first obtain a style- and domain-invariant representation for each $\mathbf{x}_i \in B$, and then transform this representation accordingly to the desired output domain $\mathbf{D}_{l_i^O}$ and style image sample \mathbf{x}_i^O . The next subsections show the details of the proposed architecture.

3.2.2 Encoder

The encoder \mathcal{E} is composed of a sequence of standard *Convolution* _{$k \times k$} —*Normalization*—*ReLU*—*Average Pooling* blocks and some *ResBlocks* (more details in the Supplementary Material), in which we replace the common *BN* layers [19] with our proposed normalization modules, which are detailed below.

Obtaining Style-Invariant Representations In the first two blocks of \mathcal{E} we whiten the first- and second-order statistics of the low-level features of each $X_i \subseteq B$, which are mainly responsible for the *style* of an image [10]. To do so, we propose the *Instance Whitening Transform (IWT)*, where the term *instance* is inspired by Instance Normalization (*IN*) [48] and highlights that the proposed transform is applied to a set of features extracted from a single image \mathbf{x}_i . Specifically, for each $\mathbf{v}_j \in X_i$, $IWT(\mathbf{v}_j)$ is defined as:

$$IWT(\mathbf{v}_j) = WC(\mathbf{v}_j; X_i, \boldsymbol{\beta}, \boldsymbol{\Gamma}). \quad (3)$$

Note that in Eq. 3 we use X_i as the batch, where X_i contains only features of a specific image \mathbf{x}_i (Sect. 3.2.1). Moreover, each $\mathbf{v}_j \in X_i$ is extracted from the first two convolutional layers of \mathcal{E} ; thus, \mathbf{v}_j has a small receptive field. This implies that whitening is performed using an *image-specific* feature centroid $\boldsymbol{\mu}_{X_i}$ and covariance matrix $\boldsymbol{\Sigma}_{X_i}$, which represent the first and second-order statistics of the low-level features of \mathbf{x}_i . On the other hand, coloring is based on the parameters $\boldsymbol{\beta}$ and $\boldsymbol{\Gamma}$, which *do not depend* on \mathbf{x}_i or l_i . The coloring operation is the analogous of the shift-scaling per-dimension transform computed in *BN* just after feature standardization [19] and is necessary to avoid decreasing the network representation capacity [41].

Obtaining domain-invariant representations In the subsequent blocks of \mathcal{E} we whiten the first- and second-order statistics which are *domain specific*. For this operation we adopt the *Domain Whitening Transform (DWT)* proposed in [37]. Specifically, for each $X_i \subseteq B$, let l_i be its domain label (Sect. 3.1) and let $B_{l_i} \subseteq B$ be the subset of features which have been extracted from *all* those images in B which share *the same domain label* l_i . Then, for each $\mathbf{v}_j \in B_{l_i}$:

$$DWT(\mathbf{v}_j) = WC(\mathbf{v}_j; B_{l_i}, \boldsymbol{\beta}, \boldsymbol{\Gamma}). \quad (4)$$

Similarly to Eq. 3, Eq. 4 performs whitening using a subset of the current feature batch. Specifically, all the features in B are partitioned depending on the domain label of the image they have been extracted from, so obtaining B_1, B_2, \dots , etc., where all the features in B_l belong to images of the domain \mathcal{D}_l . Then, B_l is used to compute domain-dependent first- and second-order statistics $(\boldsymbol{\mu}_{B_l}, \boldsymbol{\Sigma}_{B_l})$. These statistics are used to project each $\mathbf{v}_j \in B_l$ onto a domain-invariant spherical distribution. A similar idea was recently proposed

in [37] in a discriminative network for single-source UDA. However, differently from [37], we also use coloring by re-projecting the whitened features onto a new space governed by a learned multivariate distribution. This is done using the (layer-specific) parameters $\boldsymbol{\beta}$ and $\boldsymbol{\Gamma}$ which do not depend on l_i .

3.2.3 Decoder

Our decoder \mathcal{D} is functionally and structurally symmetric with respect to \mathcal{E} : it takes as input the domain- and style-invariant features computed by \mathcal{E} and projects these features onto the desired domain $\mathcal{D}_{l_i}^O$ with the style extracted from the reference image \mathbf{x}_i^O .

Similarly to \mathcal{E} , \mathcal{D} is a sequence of *ResBlocks* and a few *Upsampling*—*Normalization*—*ReLU*—*Convolution* _{$k \times k$} blocks (more details in the Supplementary Material). Similarly to Sect. 3.2.2, in the *Normalization* layers we replace *BN* with our proposed feature normalization approaches, which are detailed below.

Projecting Features onto a Domain-specific Distribution Apart from the last two blocks of \mathcal{D} (see below), all the other blocks are dedicated to project the current set of features onto a domain-specific subspace. This subspace is learned from data using domain-specific coloring parameters $(\boldsymbol{\beta}_l, \boldsymbol{\Gamma}_l)$, where l is the label of the corresponding domain. To this purpose we introduce the *conditional Domain Whitening Transform (cDWT)*, where the term “conditional” specifies that the coloring step is conditioned on the domain label l . In more detail: Similarly to Eq. 4, we first partition B into B_1, B_2, \dots , etc. However, the membership of $\mathbf{v}_j \in B$ to B_l is decided taking into account the *desired output* domain label l_i^O for each image rather than its original domain as in case of Eq. 4. Specifically, if $\mathbf{v}_j \in X_i$ and the output domain label of X_i is l_i^O , then \mathbf{v}_j is included in $B_{l_i^O}$. Once B has been partitioned, we define *cDWT* as follows:

$$cDWT(\mathbf{v}_j) = WC(\mathbf{v}_j; B_{l_i^O}, \boldsymbol{\beta}_{l_i^O}, \boldsymbol{\Gamma}_{l_i^O}). \quad (5)$$

Note that, after whitening, and differently from Eq. 4, coloring in Eq. 5 is performed using *domain-specific* parameters $(\boldsymbol{\beta}_{l_i^O}, \boldsymbol{\Gamma}_{l_i^O})$.

Applying a Specific Style In order to apply a specific style to \mathbf{x}_i , we first extract the output style from the reference image \mathbf{x}_i^O associated with \mathbf{x}_i (Sect. 3.1). This is done using the *Style Path* (Fig. 1), which consists of two *Convolution* _{$k \times k$} —*IWT*—*ReLU*—*Average Pooling* blocks (which share the parameters with the first two layers of the encoder) and a multilayer perceptron (MLP) \mathcal{F} . Following [10] we represent a style using the first- and the second-order statistics $\boldsymbol{\mu}_{X_i^O}, \mathbf{W}_{X_i^O}^{-1}$, which are extracted using the *IWT* blocks (Sect. 3.2.2). Then, we use \mathcal{F} to adapt these statistics to the

domain-specific representation obtained as the output of the previous step. In fact, in principle, for each $\mathbf{v}_j \in X_i^O$, the *Whitening()* operation inside the *IWT* transform could be “inverted” using:

$$\text{Coloring}(\mathbf{v}_j; \boldsymbol{\mu}_{X_i^O}, \mathbf{W}_{X_i^O}^{-1}). \tag{6}$$

Indeed, the coloring operation (Eq. 1) is the inverse of whitening (Eq. 2). However, the elements of X_i now lie in a feature space which is different from the output space of Eq. 3; thus, the transformation defined by *Style Path* needs to be adapted. For this reason, we use a MLP (\mathcal{F}) which implements this adaptation:

$$[\boldsymbol{\beta}_i \parallel \boldsymbol{\Gamma}_i] = \mathcal{F}([\boldsymbol{\mu}_{X_i^O} \parallel \mathbf{W}_{X_i^O}^{-1}]). \tag{7}$$

Note that, in Eq. 7, $[\boldsymbol{\mu}_{X_i^O} \parallel \mathbf{W}_{X_i^O}^{-1}]$ is the (concatenated) input and $[\boldsymbol{\beta}_i \parallel \boldsymbol{\Gamma}_i]$ is the MLP output, one input–output pair per image \mathbf{x}_i^O .

Once $(\boldsymbol{\beta}_i, \boldsymbol{\Gamma}_i)$ have been generated, we use them as the coloring parameters of our *Adaptive IWT (AdaIWT)*:

$$\text{AdaIWT}(\mathbf{v}_j) = \text{WC}(\mathbf{v}_j; X_i^O, \boldsymbol{\beta}_i, \boldsymbol{\Gamma}_i). \tag{8}$$

Equation 8 imposes style-specific first- and second-order statistics to the features of the last blocks of \mathcal{D} in order to mimic the style of \mathbf{x}_i^O .

3.3 Network training

GAN training For the sake of clarity, in the rest of the paper we use a simplified notation for \mathcal{G} , in which \mathcal{G} takes as input only one image instead of a batch. Specifically, let $\hat{\mathbf{x}}_i = \mathcal{G}(\mathbf{x}_i, l_i, l_i^O, \mathbf{x}_i^O)$ be the generated image, starting from \mathbf{x}_i ($\mathbf{x}_i \in \mathbf{D}_i$) and with desired output domain l_i^O and style image \mathbf{x}_i^O . \mathcal{G} is trained using the combination of three different losses, with the goal of changing the style and the domain of \mathbf{x}_i while preserving its content.

First, we use an *adversarial loss* based on the Projection Discriminator [30] (\mathcal{D}_P), which is conditioned on the input labels (i.e., domain labels, in our case) and uses a hinge loss:

$$\mathcal{L}_{cGAN}(\mathcal{G}) = -\mathcal{D}_P(\hat{\mathbf{x}}_i, l_i^O) \tag{9}$$

$$\begin{aligned} \mathcal{L}_{cGAN}(\mathcal{D}_P) = & \max(0, 1 + \mathcal{D}_P(\hat{\mathbf{x}}_i, l_i^O)) \\ & + \max(0, 1 - \mathcal{D}_P(\mathbf{x}_i, l_i)) \end{aligned} \tag{10}$$

The second loss is the *Identity loss* proposed in [54]), which in our framework is implemented as follows:

$$\mathcal{L}_{ID}(\mathcal{G}) = \|\mathcal{G}(\mathbf{x}_i, l_i, l_i, \mathbf{x}_i) - \mathbf{x}_i\|_1. \tag{11}$$

In Eq. 11, \mathcal{G} computes an identity transformation, being the input and the output domain and style the same. After that, a pixel-to-pixel \mathcal{L}_1 norm is computed.

Finally, we propose to use a third loss which is based on the rationale that the generation process should be *equivariant* with respect to a set of simple transformations which preserve the main content of the images (e.g., the foreground object shape). Specifically, we use the set of the affine transformations $\{h(\mathbf{x}; \boldsymbol{\theta})\}$ of image \mathbf{x} which are defined by the parameter $\boldsymbol{\theta}$ ($\boldsymbol{\theta}$ is a 2D transformation matrix). The affine transformation is implemented by a differentiable bilinear kernel as in [21]. The *equivariance loss* is:

$$\mathcal{L}_{Eq}(\mathcal{G}) = \|\mathcal{G}(h(\mathbf{x}_i; \boldsymbol{\theta}_i), l_i, l_i^O, \mathbf{x}_i^O) - h(\hat{\mathbf{x}}_i; \boldsymbol{\theta}_i)\|_1. \tag{12}$$

In Eq. 12, for a given image \mathbf{x}_i , we randomly choose a geometric parameter $\boldsymbol{\theta}_i$ and we apply $h(\cdot; \boldsymbol{\theta}_i)$ to $\hat{\mathbf{x}}_i = \mathcal{G}(\mathbf{x}_i, l_i, l_i^O, \mathbf{x}_i^O)$. Then, using the same $\boldsymbol{\theta}_i$, we apply $h(\cdot; \boldsymbol{\theta}_i)$ to \mathbf{x}_i and we get $\mathbf{x}'_i = h(\mathbf{x}_i; \boldsymbol{\theta}_i)$, which is input to \mathcal{G} in order to generate a second image. The two generated images are finally compared using the \mathcal{L}_1 norm. This is a form of self-supervision, in which equivariance to geometric transformations is used to extract semantics. Very recently a similar loss has been proposed in [18], where equivariance to affine transformations is used for image co-segmentation.

The complete loss for \mathcal{G} is:

$$\mathcal{L}(\mathcal{G}) = \mathcal{L}_{cGAN}(\mathcal{G}) + \lambda(\mathcal{L}_{Eq}(\mathcal{G}) + \mathcal{L}_{ID}(\mathcal{G})). \tag{13}$$

Note that Eqs. 9, 10 and 12 depend on the pair (\mathbf{x}_i, l_i^O) : This means that the supervisory information we effectively use, grows with $O((N+1)^2)$, which is quadratic with respect to a plain strategy in which N different source-to-target generators are trained (Sect. 2).

Classifier training Once \mathcal{G} is trained, we use it to artificially create a labeled training dataset (T^L) for the target domain. Specifically, for each S_j and each $(\mathbf{x}_i, y_i) \in S_j$, we randomly pick $\mathbf{x}_t \in T$, which is used as the reference style image, and we generate: $\hat{\mathbf{x}}_i = \mathcal{G}(\mathbf{x}_i, l_i, N+1, \mathbf{x}_t)$, where $N+1$ is fixed and indicates the target domain (\mathbf{D}_t) label (Sect. 3.1). $(\hat{\mathbf{x}}_i, y_i)$ is added to T^L and the process is iterated. T^L is generated on the fly during the training of \mathcal{C} , and, every time that a given $(\mathbf{x}_i, y_i) \in S_j$ is selected, we randomly select a different reference style image $\mathbf{x}_t \in T$.

Finally, we train a classifier \mathcal{C} on T^L using the cross-entropy loss:

$$\mathcal{L}_{Cls}(\mathcal{C}) = -\frac{1}{|T^L|} \sum_{(\hat{\mathbf{x}}_i, y_i) \in T^L} \log p(y_i | \hat{\mathbf{x}}_i). \tag{14}$$

4 Experimental results

In this section we describe the experimental setup and then we evaluate our approach using common MSDA datasets. We also present an ablation study in which we separately analyze the impact of each TriGAN component. In the Supplementary Material we show additional experiments in a single-source UDA scenario.

4.1 Datasets

In our experiments we consider two common domain adaptation benchmarks, namely the Digits-Five benchmark [50] and the Office-Caltech dataset [11].

Digits-Five [50] is composed of five different digit-recognition datasets: USPS [7], MNIST [23], MNIST-M [8], SVHN [33] and the Synthetic numbers dataset [9] (SYNDIGITS). SVHN [33] contains Google Street View images of real-world house numbers. Synthetic numbers [9] includes 500K computer-generated digits with different sources of variations (i.e., position, orientation, color, blur). USPS [7] is a dataset of digits scanned from U.S. envelopes, MNIST [23] is a popular benchmark for digit recognition and MNIST-M [8] is its colored counterpart. We adopt the experimental protocol described in [50]: in each domain the train/test split is composed of a subset of 25,000 images for training and 9000 images for testing. For USPS, the entire dataset is used.

Office-Caltech [11] is a domain adaptation benchmark, obtained selecting the subset of those 10 categories which are shared between Office31 and Caltech256 [13]. It contains 2533 images, about half of which belonging to Caltech256. There are four different domains: Amazon (A), DSLR (D), Webcam (W) and Caltech256 (C).

4.2 Experimental setup

For lack of space, we provide the architectural details of our generator \mathcal{G} and discriminator $\mathcal{D}_{\mathcal{P}}$ in the Supplementary Material. We train TriGAN for 100 epochs using the Adam optimizer [22] with the learning rate set to $1e-4$ for \mathcal{G} and $4e-4$ for $\mathcal{D}_{\mathcal{P}}$ as in [15]. The loss weighing factor λ in Eq. 13 is set to 10 as in [54].

In the Digits-Five experiments we use a mini-batch of size 256. Due to the difference in image resolution and image channels, the images of all the domains are converted to 32×32 RGB. For a fair comparison, for the final target classifier \mathcal{C} we use exactly the same network architecture used in [9,34].

In the Office-Caltech10 experiments we downsample the images to 164×164 to accommodate more samples in a mini-batch. We use a mini-batch of size 24 for training with 1 GPU. For the backbone target classifier \mathcal{C} we use the ResNet101 [14] architecture used in [34]. The weights are initialized with a network pre-trained on the ILSVRC-

2012 dataset [38]. In our experiments we remove the output layer and we replace it with a randomly initialized fully connected layer with 10 logits, one for each class of the Office-Caltech10 dataset. \mathcal{C} is trained with Adam with an initial learning rate of $1e-5$ for the randomly initialized last layer and $1e-6$ for all other layers. In the Office-Caltech10 experiments, we also include $\{S_j\}_{j=1}^N$ in T^L when training \mathcal{C} .

4.3 Results

In this section we quantitatively analyze TriGAN (Sects. 4.3.1, 4.3.2 and 4.3.3) and we show some qualitative image translation results (Sect. 4.3.4).

4.3.1 Comparison with state-of-the-art methods

Table 1 and Table 2 show the results on the Digits-Five and the Office-Caltech10 dataset, respectively. Table 1 shows that TriGAN achieves an average accuracy of 90.08% which is higher than all other methods. M^3 SDA is better than TriGAN in the **mm, up, sv, sy** \rightarrow **mt** and in the **mt, mm, sv, sy** \rightarrow **up** settings, where TriGAN is the second best. In all the other settings, TriGAN outperforms all the other approaches. As an example, in the **mt, up, sv, sy** \rightarrow **mm** setting, TriGAN is better than the second best method, M^3 SDA, by a significant margin of 10.38%. In the same table we also show the results obtained when we replace TriGAN with StarGAN [4], which is another “universal” image translator (Sect. 2). Specifically, we use StarGAN to generate synthetic target images and then we train the target classifier using the same protocol described in Sect. 3.3. The corresponding results in Table 1 show that StarGAN, despite to be known to work well for aligned face translation, drastically fails when used in this UDA scenario.

Finally, we also use Office-Caltech10, which is considered to be difficult for generative-based UDA methods because of the high-resolution images. Although the dataset is quite saturated, TriGAN achieves a classification accuracy of 97.0%, outperforming all the other methods and beating the previous state-of-the-art approach (M^3 SDA) by a margin of 0.6% on average (Table 2).

4.3.2 Ablation study

In this section we analyze the different components of our method and study in isolation their impact on the final accuracy. Specifically, we use the Digits-Five dataset and the following models: (i) Model **A**, which is our full model containing the following components: IWT , DWT , $cDWT$, $AdaIWT$ and \mathcal{L}_{Eq} . (ii) Model **B**, which is similar to Model **A** except we replace \mathcal{L}_{Eq} with the cycle-consistency loss \mathcal{L}_{Cycle} of CycleGAN [54]. (iii) Model **C**, where we replace

Table 1 Classification accuracy (%) on **Digits-Five**. *MNIST-M*, *MNIST*, *USPS*, *SVHN* and *Synthetic Digits* are abbreviated as **mm**, **mt**, **up**, **sv** and **sy**, respectively. Each setting is denoted as a set of source domains (before the arrow) and a target domain (after the arrow). Following [34], we indicate with “Source Combine” the protocol in which all the source datasets are combined in one, hence performing a standard single-source domain adaptation task. Best values are in bold, and second best values are underlined

Protocol	Models	mt, up, sv, sy → mm	mm, up, sv, sy → mt	mt, mm, sv, sy → up	mt, up, mm, sy → sv	mt, up, sv, mm → sy	Avg
Source Combine	Source Only	63.70 ± 0.83	92.30 ± 0.91	90.71 ± 0.54	71.51 ± 0.75	83.44 ± 0.79	80.33 ± 0.76
	DAN [26]	67.87 ± 0.75	97.50 ± 0.62	93.49 ± 0.85	67.80 ± 0.84	86.93 ± 0.93	82.72 ± 0.79
	DANN [8]	70.81 ± 0.94	97.90 ± 0.83	93.47 ± 0.79	68.50 ± 0.85	87.37 ± 0.68	83.61 ± 0.82
Multi-Source	Source Only	63.37 ± 0.74	90.50 ± 0.83	88.71 ± 0.89	63.54 ± 0.93	82.44 ± 0.65	77.71 ± 0.81
	DAN [26]	63.78 ± 0.71	96.31 ± 0.54	94.24 ± 0.87	62.45 ± 0.72	85.43 ± 0.77	80.44 ± 0.72
	CORAL [42]	62.53 ± 0.69	97.21 ± 0.83	93.45 ± 0.82	64.40 ± 0.72	82.77 ± 0.69	80.07 ± 0.75
	DANN [8]	71.30 ± 0.56	97.60 ± 0.75	92.33 ± 0.85	63.48 ± 0.79	85.34 ± 0.84	82.01 ± 0.76
	ADDA [46]	71.57 ± 0.52	97.89 ± 0.84	92.83 ± 0.74	75.48 ± 0.48	86.45 ± 0.62	84.84 ± 0.64
	DCTN [50]	70.53 ± 1.24	96.23 ± 0.82	92.81 ± 0.27	77.61 ± 0.41	86.77 ± 0.78	84.79 ± 0.72
	M ³ SDA [34]	72.82 ± 1.13	98.43 ± 0.68	96.14 ± 0.81	81.32 ± 0.86	89.58 ± 0.56	87.65 ± 0.75
StarGAN [4]	44.71 ± 1.39	96.26 ± 0.62	55.32 ± 3.71	58.93 ± 1.95	63.36 ± 2.41	63.71 ± 2.01	
TriGAN (ours)	83.20 ± 0.78	97.20 ± 0.45	94.08 ± 0.92	85.66 ± 0.79	90.30 ± 0.57	90.08 ± 0.70	

Table 2 Classification accuracy (%) on **Office-Caltech10**. The target domains are indicated as follows: Amazon (A), DSLR (D), Webcam (W) and Caltech256 (C). In each setting, the source domains are all the remaining datasets except the target dataset

Protocol	Models	All/W → W	All/D → D	All/C → C	All/A → A	Avg
Source Combine	Source only	99.0	98.3	87.8	86.1	92.8
	DAN [26]	99.3	98.2	89.7	94.8	95.5
Multi-Source	Source only	99.1	98.2	85.4	88.7	92.9
	DAN [26]	99.5	99.1	89.2	91.6	94.8
	DCTN [50]	99.4	99.0	90.2	92.7	95.3
	M ³ SDA [34]	<u>99.5</u>	<u>99.2</u>	<u>92.2</u>	<u>94.5</u>	<u>96.4</u>
	StarGAN [4]	99.6	100.0	89.3	93.3	95.5
	TriGAN (ours)	99.7	100.0	93.0	95.2	97.0

IWT, *DWT*, *cDWT* and *AdaIWT* of Model **A** with *IN* [48], *BN* [19], conditional Batch Normalization (*cBN*) [6] and Adaptive Instance Normalization (*AdaIN*) [17]. This comparison highlights the difference between feature whitening and feature standardization. (iv) Model **D**, which ignores the style factor. Specifically, in Model **D**, the blocks related to the style factor, i.e., the *IWT* and the *AdaIWT* blocks, are replaced by *DWT* and *cDWT* blocks, respectively. (v) Model **E**, in which the style path differs from Model **A** in the way the style is applied to the domain-specific representation. Specifically, we remove the MLP $\mathcal{F}(\cdot)$ and we directly apply $(\mu_{X_i^o}, W_{X_i^o}^{-1})$. (vi) Finally, Model **F** represents no-domain assumption (e.g., the *DWT* and *cDWT* blocks are replaced with standard WC blocks).

The results, reported in Table 3, show that all the components of the proposed generator play a role in the accuracy reached by the full model (**A**). Specifically, \mathcal{L}_{Cycle} (Model **B**) is detrimental for the accuracy because \mathcal{G} may focus on semantically meaningless details when reconstructing back the image. Conversely, the affine transformations used in case of \mathcal{L}_{Eq} , force \mathcal{G} to focus on the shape (i.e., the content) of the images. Also Model **C** is outperformed by model **A**, demonstrating the importance of feature whitening with respect to feature standardization, corroborating the findings of [37] in a pure discriminative scenario. Moreover, the no-style assumption in Model **D** hurts the classification accuracy by a margin of 1.76% when compared with Model **A**. We believe this is due to the fact that when instance-specific style information is missing in the image translation process, then the diversity of the translations decreases, consequently reducing the final accuracy. In fact, we remind that, when T^L is created (Sect. 3.3), for the same $(\mathbf{x}_i, y_i) \in S_j$, we randomly select multiple, different reference style images $\mathbf{x}_i \in T$: this diversity cannot be obtained if only domain-specific latent factors are modeled. Finally, Model **E** and Model **F** show the importance of the proposed style path and the domain factor, respectively.

Note that the ablation analysis in Table 3 is done by removing a single component from the full model **A**, and the marginal differences with respect to Model **A** show that all the components are important. On the other hand, simultaneously removing all the components makes our model similar to StarGAN, which we also report in Table 3 as a baseline comparison. In fact, in StarGAN, there is no style information and the domain labels are concatenated with the input image (Sect. 2). Conversely, our main contribution is the generation of intermediate-invariant features and their re-projection onto the target domain-and-style distribution. This is obtained using a combination of blocks which extend *WC* [41] in order to first remove and then re-introduce style- and domain-specific statistics. The difference between the use of StarGAN and the use of Model **A** to populate of T^L (-26.37%) empirically shows that the proposed image translation approach is effective in a MSDA scenario.

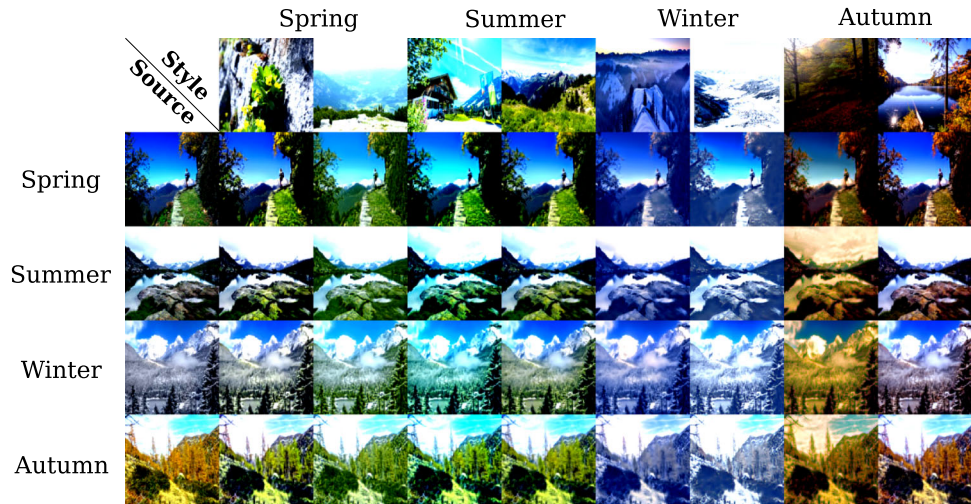
4.3.3 Multi-domain image-to-image translation

Our proposed generator can be used for a pure generative (non-UDA), multi-domain image-to-image translation task. We conduct experiments on the Alps Seasons dataset [1] which consists of images of Alps mountains with four different domains (corresponding to four seasons). Figure 2 shows some images generated using our generator. For this experiment, we compare our generator with StarGAN [4] using the FID [15] metrics. FID measures the realism of the generated images (the lower the better). The FID scores are computed considering all the real samples in the target domain and generating an equivalent number of synthetic images in the target domain. Table 4 shows that the TriGAN FID scores are significantly lower than the StarGAN scores. This further highlights that decoupling the style and the domain and using *WC*-based layers to progressively “whiten” and “color” the image statistics yields a more realistic cross-domain image translation than using domain labels as input as in the case of StarGAN.

Table 3 An analysis of the main TriGAN components using Digits-Five

Model	Description	Avg. accuracy (%) (difference)
A	TriGAN (full method)	90.08
B	Replace equivariance loss with cycle loss	88.38 (−1.70)
C	Replace whitening with feature standardization	89.39 (−0.68)
D	No style assumption	88.32 (−1.76)
E	Applying style directly instead of style path	88.36 (−1.71)
F	No domain assumption	89.10 (−0.98)
StarGAN (Baseline)	No style assumption, domain labels concatenated with the input image	63.71 (−26.37)

Fig. 2 Image translation examples generated by TriGAN across different domains (i.e., seasons) using the Alps Seasons dataset. We show two generated images for each domain combination. The leftmost column shows the *source* images, one from each domain, and the topmost row shows the *style* of the target domain, two reference style images for each target domain

**Table 4** Alps Seasons dataset, FID scores: comparing TriGAN with StarGAN [4]

	All/Wint. → Winter	All/Sum. → Summer	All/Spr. → Spring	All/Aut. → Autumn
StarGAN [4]	148.45	180.36	175.40	145.24
TriGAN (ours)	41.03	38.59	40.75	32.71

4.3.4 Qualitative image translation results

In Figs. 3 and 4, we show some translations examples obtained using our generator \mathcal{G} and different datasets, which show how both the domain and the style are used in transforming an image sample. For instance, the fourth row of Fig. 3 shows an SVHN digit image which contains other digits in the background (which is a common characteristic of the SVHN dataset). When \mathcal{G} translates this image to MNIST or MNIST-M, the background digit disappears, accordingly to the common uniform background of the target datasets. When the reference style image is, e.g., the MNIST-M “three” (fifth column), \mathcal{G} correctly applies the instance-specific style (i.e., a blue foreground digit with a red background). A similar behavior can be observed in Fig. 4.

5 Conclusions

In this work we proposed TriGAN, an MSDA framework which is based on data generation from multiple source domains using a single generator. The underlying principle of our approach is to obtain intermediate-, domain- and style-invariant representations in order to simplify the generation process. Specifically, our generator progressively removes style- and domain-specific statistics from the source images and then re-projects the intermediate features onto the desired target domain and style. We obtained state-of-the-art results on two MSDA datasets, showing the potentiality of our approach.

Fig. 3 Image translation examples obtained using our generator with the Digits-Five dataset. The leftmost column shows the *source* images, one from each domain, and the topmost row shows the *style* image from the target domain, two reference images for each target domain

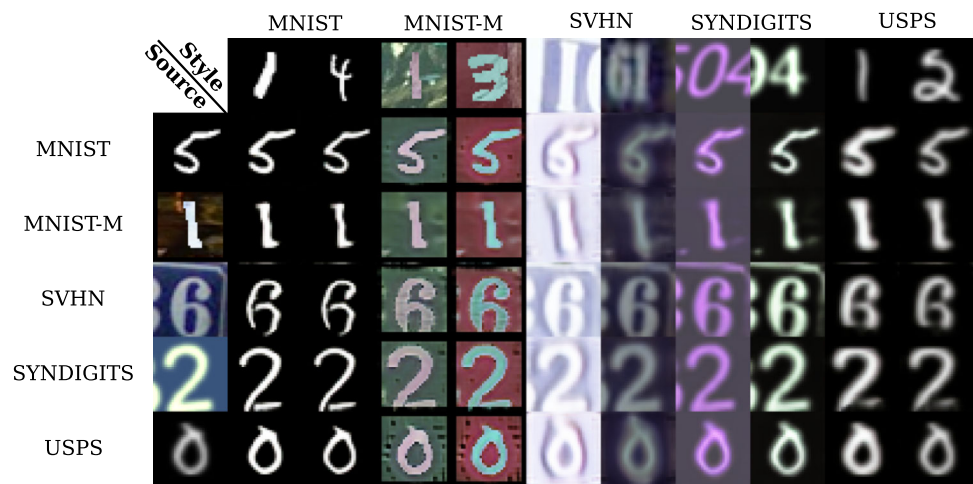


Fig. 4 Image translation examples obtained using our generator with the Office-Caltech10 dataset. The leftmost column shows the *source* images, one from each domain, and the topmost row shows the *style* image from the target domain, two reference images for each target domain



Funding Open Access funding provided by Università degli Studi di Trento

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Anoosheh, A., Agustsson, E., Timofte, R., Van Gool, L.: Com-bogan: unrestrained scalability for image domain translation. In: CVPR (2018)
- Carlucci, F.M., Porzi, L., Caputo, B., Ricci, E., Bulò, S.R.: Auto-dial: automatic domain alignment layers. In: ICCV (2017)
- Cho, W., Choi, S., Park, D.K., Shin, I., Choo, J.: Image-to-image translation via group-wise deep whitening-and-coloring transformation. In: CVPR (2019)
- Choi, Y., Choi, M., Kim, M., Ha, J.W., Kim, S., Choo, J.: StarGAN: unified generative adversarial networks for multi-domain image-to-image translation. In: CVPR (2018)
- Dereniowski, D., Kubale, M.: Cholesky factorization of matrices in parallel and ranking of graphs. In: International Conference on Parallel Processing and Applied Mathematics (2003)
- Dumoulin, V., Shlens, J., Kudlur, M.: A learned representation for artistic style. In: ICLR (2016)
- Friedman, J., Hastie, T., Tibshirani, R.: The elements of statistical learning, vol. 1. Springer series in statistics New York (2001)
- Ganin, Y., Lempitsky, V.: Unsupervised domain adaptation by backpropagation. In: ICML (2015)
- Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., Lempitsky, V.: Domain-adversarial training of neural networks. JMLR (2016)
- Gatys, L.A., Ecker, A.S., Bethge, M.: Image style transfer using convolutional neural networks. In: CVPR (2016)
- Gong, B., Shi, Y., Sha, F., Grauman, K.: Geodesic flow kernel for unsupervised domain adaptation. In: CVPR (2012)

12. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: NIPS (2014)
13. Griffin, G., Holub, A., Perona, P.: Caltech-256 object category dataset (2007)
14. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)
15. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: Gans trained by a two time-scale update rule converge to a local nash equilibrium. In: NIPS (2017)
16. Hoffman, J., Tzeng, E., Park, T., Zhu, J.Y., Isola, P., Saenko, K., Efros, A.A., Darrell, T.: Cycada: Cycle-consistent adversarial domain adaptation. In: ICML (2017)
17. Huang, X., Liu, M.Y., Belongie, S., Kautz, J.: Multimodal unsupervised image-to-image translation. In: ECCV (2018)
18. Hung, W.C., Jampani, V., Liu, S., Molchanov, P., Yang, M.H., Kautz, J.: Scops: self-supervised co-part segmentation. In: CVPR (2019)
19. Ioffe, S., Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate shift. In: ICML (2015)
20. Isola, P., Zhu, J.Y., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. In: CVPR (2017)
21. Jaderberg, M., Simonyan, K., Zisserman, A., et al.: Spatial transformer networks. In: NIPS (2015)
22. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. [arXiv:1412.6980](https://arxiv.org/abs/1412.6980) (2014)
23. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. In: Proceedings of the IEEE (1998)
24. Li, Y., Wang, N., Shi, J., Liu, J., Hou, X.: Revisiting batch normalization for practical domain adaptation. In: ICLR-WS (2017)
25. Liu, M.Y., Tuzel, O.: Coupled generative adversarial networks. In: NIPS (2016)
26. Long, M., Wang, J.: Learning transferable features with deep adaptation networks. In: ICML (2015)
27. Long, M., Zhu, H., Wang, J., Jordan, M.I.: Deep transfer learning with joint adaptation networks. In: ICML (2017)
28. Mancini, M., Porzi, L., Bulò, S.R., Caputo, B., Ricci, E.: Boosting domain adaptation by discovering latent domains. In: CVPR (2018)
29. Mansour, Y., Mohri, M., Rostamizadeh, A.: Domain adaptation with multiple sources. In: NIPS (2009)
30. Miyato, T., Koyama, M.: cGANs with projection discriminator. In: ICLR (2018)
31. Morerio, P., Cavazza, J., Murino, V.: Minimal-entropy correlation alignment for unsupervised deep domain adaptation. In: ICLR (2018)
32. Murez, Z., Kolouri, S., Kriegman, D., Ramamoorthi, R., Kim, K.: Image to image translation for domain adaptation. In: CVPR (2018)
33. Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A.Y.: Reading digits in natural images with unsupervised feature learning. In: NIPS-WS (2011)
34. Peng, X., Bai, Q., Xia, X., Huang, Z., Saenko, K., Wang, B.: Moment matching for multi-source domain adaptation. ICCV (2019)
35. Peng, X., Saenko, K.: Synthetic to real adaptation with generative correlation alignment networks. In: WACV (2018)
36. Rakshit, S., Banerjee, B., Roig, G., Chaudhuri, S.: Unsupervised multi-source domain adaptation driven by deep adversarial ensemble learning. In: DAGM (2019)
37. Roy, S., Siarohin, A., Sangineto, E., Bulò, S.R., Sebe, N., Ricci, E.: Unsupervised domain adaptation using feature-whitening and consensus loss. CVPR (2019)
38. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: Imagenet large scale visual recognition challenge. IJCV (2015)
39. Russo, P., Carlucci, F.M., Tommasi, T., Caputo, B.: From source to target and back: symmetric bi-directional adaptive gan. In: CVPR (2018)
40. Sankaranarayanan, S., Balaji, Y., Castillo, C.D., Chellappa, R.: Generate to adapt: aligning domains using generative adversarial networks. In: CVPR (2018)
41. Siarohin, A., Sangineto, E., Sebe, N.: Whitening and Coloring batch transform for GANs. In: ICLR (2019)
42. Sun, B., Feng, J., Saenko, K.: Return of frustratingly easy domain adaptation. In: AAAI (2016)
43. Sun, B., Saenko, K.: Deep coral: Correlation alignment for deep domain adaptation. In: ECCV (2016)
44. Torralba, A., Efros, A.A.: Unbiased look at dataset bias. In: CVPR (2011)
45. Tzeng, E., Hoffman, J., Darrell, T., Saenko, K.: Adversarial discriminative domain adaptation. In: CVPR (2017)
46. Tzeng, E., Hoffman, J., Saenko, K., Darrell, T.: Adversarial discriminative domain adaptation. In: CVPR (2017)
47. Tzeng, E., Hoffman, J., Zhang, N., Saenko, K., Darrell, T.: Deep domain confusion: Maximizing for domain invariance. [arXiv:1412.3474](https://arxiv.org/abs/1412.3474) (2014)
48. Ulyanov, D., Vedaldi, A., Lempitsky, V.S.: Instance normalization: the missing ingredient for fast stylization. [arXiv:1607.08022](https://arxiv.org/abs/1607.08022) (2016)
49. Venkateswara, H., Eusebio, J., Chakraborty, S., Panchanathan, S.: Deep hashing network for unsupervised domain adaptation. In: CVPR (2017)
50. Xu, R., Chen, Z., Zuo, W., Yan, J., Lin, L.: Deep cocktail network: Multi-source unsupervised domain adaptation with category shift. In: CVPR (2018)
51. Yao, Y., Doretto, G.: Boosting for transfer learning with multiple sources. In: CVPR (2010)
52. Zhao, S., Li, B., Yue, X., Gu, Y., Xu, P., Hu, R., Chai, H., Keutzer, K.: Multi-source domain adaptation for semantic segmentation. In: NeurIPS (2019)
53. Zhao, S., Wang, G., Zhang, S., Gu, Y., Li, Y., Song, Z., Xu, P., Hu, R., Chai, H., Keutzer, K.: Multi-source distilling domain adaptation. In: AAAI (2020)
54. Zhu, J.Y., Park, T., Isola, P., Efros, A.A.: Unpaired image-to-image translation using cycle-consistent adversarial networks. In: CVPR (2017)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.