GENERATIVE MULTI-SOURCE DOMAIN ADAPTATION

Anonymous authors
Paper under double-blind review

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 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.

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 another domain, *i.e.* the *target*. Since the two domains typically have different marginal feature distributions, the adaptation process needs to align the one to the other in order to reduce the *domain shift* (Torralba & Efros (2011)). 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) (Yao & Doretto (2010); Mansour et al. (2009); Xu et al. (2018); Peng et al. (2019)) methods may be adopted, in which more than one source dataset is considered in order to make the adaptation process more robust. However, despite more data can be used, MSDA is challenging as multiple domain shift problems need to be simultaneously and coherently solved.

In this paper we tackle MSDA (unsupervised) problem and we propose a novel Generative Adversarial Network (GAN) for addressing the domain shift when multiple source domains are available. Our solution is based on generating artificial target samples by transforming images from all the source domains. Then the synthetically generated images are used for training the target classifier. While this strategy has been recently adopted in single-source UDA scenarios (Russo et al. (2018); Hoffman et al. (2017); Liu & Tuzel (2016); Murez et al. (2018); Sankaranarayanan et al. (2018)), we are the first to show how it can be effectively exploited in a MSDA setting.

The holy grail of any domain adaptation method is to obtain domain invariant representations. Similarly, in multi-domain image-to-image translation tasks it is very crucial to obtain domain invariant representations in order to reduce the number of learned translations from $\mathcal{O}(N^2)$ to $\mathcal{O}(N)$, where N is the number of domains. Several domain adaptation methods (Roy et al. (2019); Carlucci et al. (2017); Sun & Saenko (2016); Tzeng et al. (2014)) achieve domain-invariant representations by aligning only domain specific distributions. However, we postulate that style is the most important *latent* factor that describe a domain and need to be modelled separately for obtaining optimal domain invariant representation. More precisely, in our work we assume that the appearance of an image depends on three factors: *i.e.* the *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, whereas, the factor *style* represents a property that is shared among different parts of *a single image* and describes low-level features which concern a specific image. Our generator obtains the do-

main invariant representation in a two-step process, by first obtaining style invariant representations followed by achieving domain invariant representation.

In more detail, the proposed translation is implemented using a *style-and-domain translation* generator. This generator is composed of two main components, an encoder and a decoder. Inspired by (Roy et al. (2019)) in the encoder we embed *whitening* layers that progressively align the style-and-domain feature distributions in order to obtain a representation of the image content which is invariant to these factors. Then, in the decoder, we project this invariant representation onto a new domain-and-style specific distribution with *Whitening and Coloring* (WC) (Siarohin et al. (2019)) batch transformations, according to the target data. Importantly, the use of an intermediate, explicit invariant representation, obtained through WC, makes the number of domain transformations which need to be learned linear with the number of domains. In other words, this design choice ensures scalability when the number of domains increases, which is a crucial aspect for an effective MSDA method.

Contributions. Our main contributions can be summarized as follows. (i) We propose the first generative model dealing with MSDA. We call our approach TriGAN because it is based on three different factors of the images: the style, the domain and the content. (ii) The proposed *style-and-domain translation* generator 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: Instance Whitening Transform (IWT), Domain Whitening Transform (DWT) (Roy et al. (2019)), conditional Domain Whitening Transform ($cDWT$) and Adaptive Instance Whitening Transform ($AdaIWT$). Notably, the IWT and $AdaIWT$ are novel layers introduced with this paper. (iii) We test our method on two MSDA datasets, Digits-Five (Xu et al. (2018)) and Office-Caltech10 (Gong et al. (2012)), outperforming state-of-the-art methods.

2 Related Work

In this section we review previous approaches on UDA, considering both single source and multi-source methods. Since, the proposed architecture is also related to deep models used for image-to-image translation, we also discuss 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 upon the strategies adopted to cope with the domain-shift problem. The first category utilizes the first and second order statistics to model the source and target feature distributions. For instance, (Long & Wang (2015); Long et al. (2017); Venkateswara et al. (2017); Tzeng et al. (2014)) minimize the Maximum Mean Discrepancy, *i.e.* the distance between the mean of feature distributions between the two domains. On the other hand, (Sun & Saenko (2016); Morerio et al. (2018); Peng & Saenko (2018)) achieve domain invariance by aligning the second-order statistics through correlation alignment. Differently, (Carlucci et al. (2017); Li et al. (2017); Mancini et al. (2018)) reduce the domain shift by domain alignment layers derived from batch normalization (BN) (Ioffe & Szegedy (2015)). This idea has been recently extended in (Roy et al. (2019)), where grouped-feature whitening (DWT) is used instead of feature standardization as in BN. Contrarily, in our proposed encoder we use the WC transform, 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 aim to build 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 (Ganin & Lempitsky (2015)). Similarly, the approach in (Tzeng et al. (2017)) utilizes a domain confusion loss to promote the alignment between the source and the target domain. A third category of methods use adversarial learning in a generative framework (i.e., GANs (Goodfellow et al. (2014)) to reconstruct artificial source and/or target images and perform domain adaptation. Notable approaches are SBADA-GAN (Russo et al. (2018)), CyCADA (Hoffman et al. (2017)), CoGAN (Liu & Tuzel (2016)), I2I Adapt (Murez et al. (2018)) and Generate To Adapt (GTA) (Sankaranarayanan et al. (2018)). While these generative methods have been shown to be very successful in UDA, none of them deals with a multi-source setting. Indeed, extending these approaches to deal with multiple source domains is not trivial, because the construction of $\mathcal{O}(N^2)$ one-to-one translation generators and discriminator networks would most likely dramatically increase the number of parameters which need to be trained.

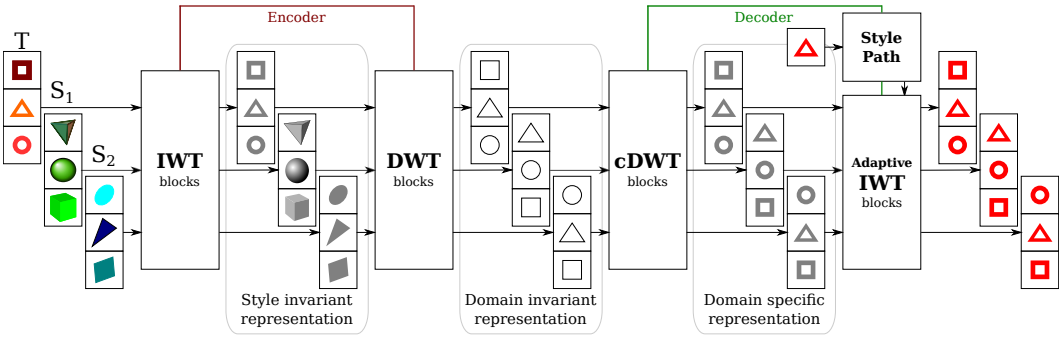


Figure 1: An overview of the TriGAN generator. We schematically show 3 domains $\{T, S_1, S_2\}$ - objects with *holes*, *3D objects* and *skewered* objects, respectively. The content is represented by the object's shape - square, circle or triangle. The style is represented by the color: each image input to $\mathcal{G}$ has a different color and each domain has it 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 representation. Symmetrically, the decoder $\mathcal{D}$ brings back domain-specific information with cDWT blocks (for simplicity we show only a single domain, T). Finally, we apply a reference style. The reference style is extracted using style path and it is applied using Adaptive IWT blocks.

Multi-source UDA. Yao & Doretto (2010) deal with multiple-source knowledge transfer by borrowing knowledge from the target *k* nearest-neighbour sources. Similarly, a distribution-weighed combining rule is proposed in (Mansour et al. (2009)) to construct a target hypothesis as a weighted combination of source hypotheses. Recently, Deep Cocktail Network (DCTN) (Xu et al. (2018)) uses the distribution-weighted combining rule in an adversarial setting. A Moment Matching Network (M^{3}SDA) is introduced in (Peng et al. (2019)) for reducing the discrepancy between the multiple source and the target domains. Differently from these methods which operate in a discriminative setting, our method relies on a deep generative approach for MSDA.

Image-to-image Translation. Image-to-image translation approaches, i.e. the methods that learn how to transform an image from one domain to another, possibly keeping its semantics, are the basis of our method. In (Isola et al. (2017)) the pix2pix network translates images under the assumption that paired images in the two domains are available at training time. In contrast, CycleGAN (Zhu et al. (2017)) can learn to translate images using unpaired training samples. Note that, by design, these methods work with two domains. ComboGAN (Anoosheh et al. (2018)) partially alleviates this issue by using N generators for translations among N domains. Our work is also related to StarGAN (Choi et al. (2018)) which handles unpaired image translation amongst *N* domains ($N \geq 2$) through a single generator. However, StarGAN achieves image translation without explicitly forcing representations to be domain invariant and this may lead to significant reduction of 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 colouring based image-to-image translation method was proposed in (Cho et al. (2019)), where the whitening operation is *weight-based*. Specifically, the whitening operation is approximated by enforcing the convariance matrix, computed from the intermediate features, to be equal to the identity matrix. Conversely, our whitening layers are *data dependent* and they use the Cholesky decomposition (Dereniowski & Kubale (2003)) to compute the whitening matrices from the input samples in a closed form, thereby eliminating the need for additional ad-hoc losses.

3 The Style-and-Domain Translation Generator

In this section we describe the proposed approach for MSDA. We first provide an overview of our method and introduce the notation adopted throughout the paper (Sec. 3.1). Then we describe the TriGAN architecture (Sec. 3.2) and our training procedure (Sec.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 categories and each of them is associated to a domain $\mathbf{D}_1^s, ..., \mathbf{D}_N^s, \mathbf{D}^t$, 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 steps. 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. Importantly, our $\mathcal{G}$ learns mappings between every possible pair of image domains. Once $\mathcal{G}$ is trained, 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 $\mathcal{C}$. In more detail, $\mathcal{G}$ is trained using $\{S_j\}_{j=1}^N \cup T$, however no class label is involved in this phase and T is treated in the same way as the other domain datasets. As mentioned in Sec. 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. Conversely and symmetrically, $\mathcal{D}$ needs to progressively project the domain-and-style invariant features generated by $\mathcal{E}$ onto a domain-and-style specific space.

At training time, $\mathcal{G}$ takes as input a batch of images $B = \{\mathbf{x}_1, ..., \mathbf{x}_m\}$ with corresponding domain labels $L = \{l_1, ..., l_m\}$, where $\mathbf{x}_i$ belongs to the domain $\mathbf{D}_{l_i}$ and $l_i \in [1, N+1]$. Moreover, $\mathcal{G}$ takes as input a batch of output domain labels $L^O = \{l_1^O, ..., l_m^O\}$, and a batch of style images $B^O = \{\mathbf{x}_1^O, ..., \mathbf{x}_m^O\}$, 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 image $\mathbf{x}_i^O$.

3.2 TRIGAN ARCHITECTURE

The TriGAN architecture is made of a generator network $\mathcal{G}$ and a discriminator $\mathcal{D}_\mathcal{P}$. As stated above, $\mathcal{G}$ comprises an encoder $\mathcal{E}$ and decoder $\mathcal{D}$, which will be described in (Sec. 3.2.2-3.2.3). The discriminator $\mathcal{D}_\mathcal{P}$ is based on the Projection Discriminator (Miyato & Koyama (2018)). Before describing the details of $\mathcal{G}$, we briefly review the WC transform (Siarohin et al. (2019)) (Sec. 3.2.1) which is used as the basic operation in our proposed batch-based feature transformations.

3.2.1 PRELIMINARIES: WHITENING & 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, in this way each image $\mathbf{x}_i$ contains a set of vectors $X_i = \{\mathbf{v}_1, ..., \mathbf{v}_{h \times w}\}$. With a slight abuse of the notation, we use $B = \bigcup_{i=1}^{m} X_i = \{\mathbf{v}_1, ..., \mathbf{v}_{h \times w \times m}\}$, 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 (Ioffe & Szegedy (2015)) and widely adopted in both generative and discriminative networks. WC can be described by:

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

where:

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

In Eq. 2, $\boldsymbol{\mu}_B$ is the centroid of the elements in B, while $\boldsymbol{W}_B$ is such that: $\boldsymbol{W}_B^\top \boldsymbol{W}_B = \boldsymbol{\Sigma}_B^{-1}$, 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 $\bar{B} = \{\bar{\mathbf{v}}_1, ..., \bar{\mathbf{v}}_{h \times w \times m}\}$, 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 $\boldsymbol{W}_B$ are computed from the elements in B, Eq. 1 depends on the learnable d dimensional vector $\boldsymbol{\beta}$ and $d \times d$ dimensional matrix $\boldsymbol{\Gamma}$. Eq. 1 is a linear operation and can be simply implemented using a convolutional layer with kernel size 1×1.

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 sub-sections show the details of the proposed architecture.

3.2.2 ENCODER

The encoder $\mathcal{E}$ is composed of a sequence of standard $Convolution_{k\times k}$ - $Normalization$ - $ReLU$ - $AveragePooling$ blocks and some $ResBlocks$ (more details in Appendix B), in which we replace the common BN layers (Ioffe & Szegedy (2015)) with our proposed normalization modules, which are detailed below.

Obtaining Style Invariant Representations. In the first two blocks of $\mathcal{E}$ we whiten 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 (Gatys et al. (2016)). To do so, we propose the *Instance Whitening Transform* (IWT), where the term *instance* is inspired by Instance Normalization (IN) (Ulyanov et al. (2016)) and highlights that the proposed transform is applied to a set of features extracted from a single image $\mathbf{x}_i$. For each $\mathbf{v}_j \in X_i$, $IWT(\mathbf{v}_j)$ is defined as:

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

Eq. 3 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 $\mathbf{x}_i$. 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 (Ioffe & Szegedy (2015)) and is necessary to avoid decreasing the network representation capacity (Siarohin et al. (2019)).

Obtaining Domain Invariant Representations. In the subsequent blocks of $\mathcal{E}$ we whiten first and second-order statistics which are *domain specific*. For this operation we adopt the *Domain Whitening Transform* (DWT) proposed in (Roy et al. (2019)). Specifically, for each $X_i \subseteq B$, let l_i be its domain label (see Sec. 3.1) and let $B_{l_i} \subseteq B$ be the subset of feature which have been extracted from *all* those images in B which share *the same domain label*. Then, for each $\mathbf{v}_j \in B_{l_i}$:

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

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 $B_1, B_2, ...$, etc, where all the features in B_l belongs to the images from domain $\mathbf{D}_l$. Then, first and second order statistics $(\boldsymbol{\mu}_{B_l}, \boldsymbol{\Sigma}_{B_l})$ are computed thus effectively projecting each $\mathbf{v}_j \in B_{l_i}$ onto a domain-invariant spherical distribution. A similar idea was recently proposed in (Roy et al. (2019)) in a discriminative network for single-source UDA. However, differently from (Roy et al. (2019)), 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 learnable 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 domain and style invariant features computed by $\mathcal{E}$ and projects these features onto the desired domain $\mathbf{D}_{l_i^O}$ with style extracted from desired 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 Appendix B). Similarly to Sec. 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 $B_1, B_2, ...$, 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 to Eq. 4. Specifically, let $\mathbf{v}_j \in X_i$ and the output domain is given by the label l_i^O, then $\mathbf{v}_j$ is included in $B_{l_i^O}$. Once B has been partitioned we define $cDWT$ as follows:

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

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 given style to $\mathbf{x}_i$, we extract the style from image $\mathbf{x}_i^O$ using the *Style Path* (see Fig. 1). *Style Path* consists of two $Convolution_{k\times k}$ - IWT - $ReLU$ - $AveragePooling$ blocks (which shares the parameters with the first two layers of encoder) and a MultiLayer Perceptron (MLP) $\mathcal{F}$. Following (Gatys et al. (2016)) we describe a style using first and second order statistics $\boldsymbol{\mu}_{X_i^O}, \boldsymbol{W}_{X_i^O}^{-1}$, which are extracted using the IWT blocks. 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:

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

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 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:

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

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

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

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

Eq. 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 $\mathbf{x}_i$ ($\mathbf{x}_i \in \mathbf{D}_{l_i}$) 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 (Miyato & Koyama (2018)) ($\mathcal{D}_\mathcal{P}$), which is conditioned on labels (domain labels, in our case) and uses a hinge loss:

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

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

The second loss is the *Identity loss* proposed in (Zhu et al. (2017)), and 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 differentiable billinear kernel as in (Jaderberg et al. (2015)). 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 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 (Hung et al. (2019)), where equivariance to affine transformations is used for image co-segmentation.

The complete loss for $\mathcal{G}$ is:

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

Classifier Training. Once $\mathcal{G}$ is trained, we use it to artificially create a labeled training dataset for the target domain. Specifically, for each S_j and each $(\mathbf{x}_i, y_i) \in S_j$, we randomly pick one image $\mathbf{x}_t$ from T which is used as the style-image reference, 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 (see Sec. 3.1). $(\hat{\mathbf{x}}_i, y_i)$ is added to T^L and the process is iterated.

Finally, we train a classfier $\mathcal{C}$ 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 MSDA datasets. We also present an ablation study in which we analyse the impact of each of TriGAN component on the classification accuracy.

4.1 Datasets

In our experiments we consider two common domain adaptation benchmarks, namely the Digits-Five benchmark (Xu et al. (2018)) and the Office-Caltech (Gong et al. (2012)).

The **Digits-Five** (Xu et al. (2018)) is composed of five digit-recognition datasets: USPS (Friedman et al. (2001)), MNIST (LeCun et al. (1998)), MNIST-M (Ganin & Lempitsky (2015)), SVHN (Netzer et al. (2011)) and Synthetic numbers datasets (Ganin et al. (2016)) (SYNDIGITS). SVHN (Netzer et al. (2011)) contains images from Google Street View of real-world house numbers. Synthetic numbers (Ganin et al. (2016)) includes 500K computer-generated digits with different sources of variations (*i.e.* position, orientation, color, blur). USPS (Friedman et al. (2001)) is a dataset of digits scanned from U.S. envelopes, MNIST (LeCun et al. (1998)) is a popular benchmark for digit recognition and MNIST-M (Ganin & Lempitsky (2015)) is its colored counterpart. We adopt the experimental protocol described in (Xu et al. (2018)): in each domain the train/test split is composed of a subset of 25000 images for training and 9000 for testing. For USPS the entire dataset is used.

The **Office-Caltech** (Gong et al. (2012)) is a domain-adaptation benchmark obtained selecting the subset of 10 categories shared between the Office31 and the Caltech256 (Griffin et al. (2007)) datasets. It contains 2533 images, about half of which belong to Caltech256. There are four different domains: Amazon (A), DSLR (D), Webcam (W) and Caltech256 (C).

4.2 Experimental Setup

We provide architecture details about our generator $\mathcal{G}$ and discriminator $\mathcal{D}_\mathcal{P}$ in the Appendix B. We train TriGAN for 100 epochs using the Adam optimizer (Kingma & Ba (2014)) with the learning rate set to 1e-4 for the $\mathcal{G}$ and 4e-4 for the $\mathcal{D}_\mathcal{P}$ as in (Heusel et al. (2017)). The loss weighing factor

λ in Eqn. 13 is set to 10 as in (Zhu et al. (2017)). All other hyperparameters are chosen by cross-validating on the MNIST-M, USPS, SVHN, SYNDIGITS $\rightarrow$ MNIST adaptation setting and are used in all the other settings.

For the Digits-Five experiments we use a mini-batch of size 256 for TriGAN training. Due to the difference in image resolution and image channels, the images of all the domains are converted to 32 $\times$ 32 RGB. For a fair comparison, for the final target classifier $\mathcal{C}$ we use exactly the same network architecture used in (Ganin et al. (2016)).

In the Office-Caltech10 experiments we downsample the images to 164 $\times$ 164 to accommodate more samples in a mini-batch. We use a mini-batch of size 24 for training with 1 GPU. For the back-bone target classifier $\mathcal{C}$ we use the ResNet101 (He et al. (2016)) architecture used by Peng et al. (2019). The weights are initialized with a network pre-trained on the ILSVRC-2012 dataset (Russakovsky et al. (2015)). In our experiments we remove the output layer and we replace it with a randomly initialized fully-connected layer that has 10 logits, one per 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. Since there are only a few training data in the T^L dataset, we also use $\{S_j\}_{j=1}^N$ for training $\mathcal{C}$.

4.3 Results

In this section we analyse our proposal using an ablation study and we compare with MSDA state-of-the-art methods. In Appendix A we show our qualitative results.

4.3.1 Comparison with State-of-the-Art Methods

In this section we compare our method with previous MSDA approaches. Tab. 1 and Tab. 2 show the results on the Digits-Five and Office-Caltech10 datset, 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 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 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%. For the StarGAN (Choi et al. (2018)) baseline, synthetic images are generated in the target domain and a target classifier is trained using our protocol described in Sec. 3.3. StarGAN, despite known to work well for aligned face translation, fails drastically when digits are concerned. This shows the importance of a well-designed generator that enforces domain invariant representations in the MSDA setting when there is a significant domain shift.

Standards	Models	$\rightarrow$ **mm**	$\rightarrow$ **mt**	$\rightarrow$ **up**	$\rightarrow$ **sv**	$\rightarrow$ **sy**	**Avg**
Source Combine	Source Only	63.70	92.30	90.71	71.51	83.44	80.33
	DAN (Long & Wang (2015))	67.87	97.50	93.49	67.80	86.93	82.72
	DANN (Ganin & Lempitsky (2015))	70.81	97.90	93.47	68.50	87.37	83.61
Multi-Source	Source Only	63.37	90.50	88.71	63.54	82.44	77.71
	DAN (Long & Wang (2015))	63.78	96.31	94.24	62.45	85.43	80.44
	CORAL (Sun et al. (2016))	62.53	97.21	93.45	64.40	82.77	80.07
	DANN (Ganin & Lempitsky (2015))	71.30	97.60	92.33	63.48	85.34	82.01
	ADDA (Tzeng et al. (2017))	71.57	97.89	92.83	75.48	86.45	84.84
	DCTN (Xu et al. (2018))	70.53	96.23	92.81	77.61	86.77	84.79
	M^{3}SDA (Peng et al. (2019))	<u>72.82</u>	**98.43**	**96.14**	<u>81.32</u>	<u>89.58</u>	<u>87.65</u>
	StarGAN (Choi et al. (2018))	44.71	96.26	55.32	58.93	63.36	63.71
	TriGAN (Ours)	**83.20**	<u>97.20</u>	<u>94.08</u>	**85.66**	**90.30**	**90.08**

Table 1: Classification accuracy (%) on **Digits-Five** experiments. *MNIST-M, MNIST, USPS, SVHN, Synthetic Digits* are abbreviated as **mm**, **mt**, **up**, **sv** and **sy** respectively. The best value is in bold and the second best is underlined.

Finally, we also experimented using the Office-Caltech10, which is considered to be difficult for reconstruction-based GAN 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^3SDA) by a margin of 0.6% on average (see Tab. 2).

Standards	Models	$\rightarrow$ W	$\rightarrow$ D	$\rightarrow$ C	$\rightarrow$ A	Avg
Source Combine	Source only	99.0	98.3	87.8	86.1	92.8
	DAN (Long & Wang (2015))	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 (Long & Wang (2015))	99.5	99.1	89.2	91.6	94.8
	DCTN (Xu et al. (2018))	99.4	99.0	90.2	92.7	95.3
	M^3SDA (Peng et al. (2019))	<u>99.5</u>	<u>99.2</u>	<u>92.2</u>	<u>94.5</u>	<u>96.4</u>
	StarGAN (Choi et al. (2018))	99.6	**100.0**	89.3	93.3	95.5
	TriGAN (Ours)	**99.7**	**100.0**	**93.0**	**95.2**	**97.0**

Table 2: Classification accuracy (%) on **Office-Caltech10** dataset. ResNet-101 pre-trained on ImageNet is used as the backbone network. The best value is in bold and the second best is underlined.

4.3.2 Ablation Study

In this section we analyse 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 (Zhu et al. (2017)). iii) Model **C**, where we replace *IWT*, *DWT*, *cDWT* and *AdaIWT* of Model **A** with *IN* (Ulyanov et al. (2016)), *BN* (Ioffe & Szegedy (2015)), conditional Batch Normalization (*cBN*) (Dumoulin et al. (2016)) and Adaptive Instance Normalization (*AdaIN*) (Huang et al. (2018)). This comparison highlights the difference between feature whitening and feature standardisation. 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 $\mathcal{F}(.)$ and we directly apply $(\boldsymbol{\mu}_{X_i^O}, \boldsymbol{W}_{X_i^O}^{-1})$. vi) Finally, Model **F** represents no-domain assumption (e.g. the DWT and cDWT blocks are replaced with standard WC blocks).

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 Standardisation	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)

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

Tab. 3 shows that Model **A** outperforms all the ablated models. Model **B** shows that $\mathcal{L}_{Cycle}$ is detrimental for the accuracy because $\mathcal{G}$ may focus on meaningless information to reconstruct back the image. Conversely, the affine transformations used in case of $\mathcal{L}_{Eq}$, force $\mathcal{G}$ to focus on the shape of the content of the images. Model **C** is outperformed by model **A**, demonstrating the importance of feature whitening over feature standardisation, corroborating the findings of (Roy et al. (2019)) in a pure-discriminative setting. 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 only domain-specific latent factors are modeled but instance-specific style information is missing in the image translation process, then the diversity of translations decreases, consequently reducing the final accuracy. Model **E** shows the need of using the proposed style path. Finally, Model **F** shows that having a separate factor for domain yields better performance

4.3.3 MULTI DOMAIN IMAGE-TO-IMAGE TRANSLATION

Our proposed method can be used for multi-domain image-to-image translation tasks. We conduct experiments on Alps Seasons dataset (Anoosheh et al. (2018)) which consists of images of Alps mountain range belonging to four different domains. Fig. 2 shows some images generated using our generator on the Alps Seasons. For this experiment we compare our generator with StarGAN (Choi et al. (2018)) and report the FID (Heusel et al. (2017)) metrics for the generated images. FID measures the realism of generated images and it is desirable to have a lower FID score. The FID is computed considering all the real samples in the target domain and generating equivalent number of synthetic images in the target domain. It can be observed from Tab. 4 that the FID scores of our approach is significantly lower than that of StarGAN. This further highlights the fact that explicit enforcing of domain and style invariant representation is essential for multi-domain translation.

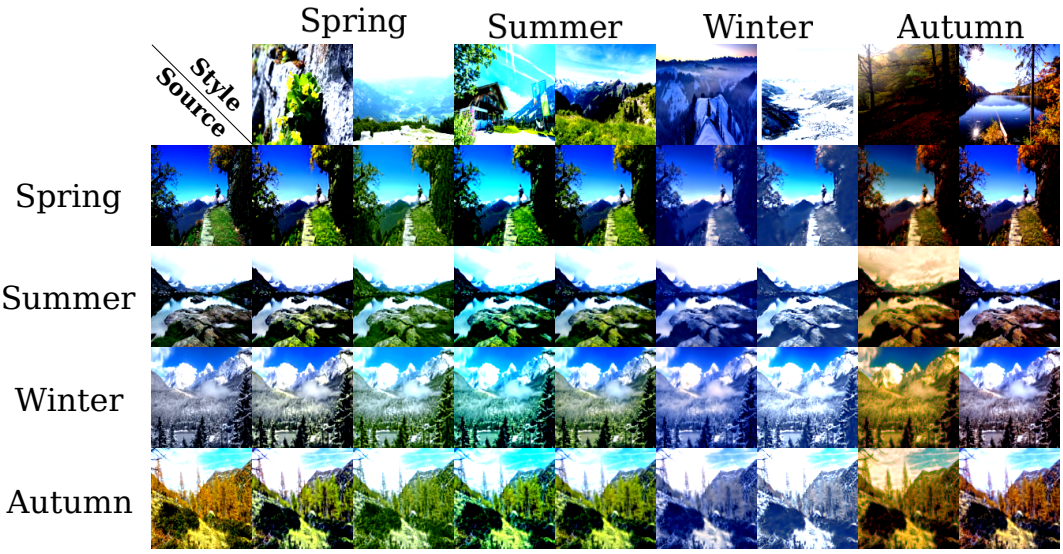


Figure 2: Image generated by TriGAN across different domains (i.e., seasons). We show two generated images for each domain combination.

	Target	→Winter	→Summer	→Spring	→Autumn
FID	StarGAN (Choi et al. (2018))	148.45	180.36	175.40	145.24
	TriGAN (Ours)	41.03	38.59	40.75	32.71

Table 4: Alps Seasons: Comparing TriGAN with StarGAN (Choi et al. (2018))

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 to to obtain domain-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 so obtained invariant representation onto the desired target domain and styles. We obtained state-of-the-art results on two MSDA datasets, showing the potentiality of our approach. We performed a detailed ablation study which shows the importance of each component of the proposed method.

REFERENCES

Asha Anoosheh, Eirikur Agustsson, Radu Timofte, and Luc Van Gool. Combogan: Unrestrained scalability for image domain translation. In *CVPR*, 2018.

Konstantinos Bousmalis, Nathan Silberman, David Dohan, Dumitru Erhan, and Dilip Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. In *CVPR*, 2017.

Fabio Maria Carlucci, Lorenzo Porzi, Barbara Caputo, Elisa Ricci, and Samuel Rota Bulò. Autodial: Automatic domain alignment layers. In *ICCV*, 2017.

Wonwoong Cho, Sungha Choi, David Keetae Park, Inkyu Shin, and Jaegul Choo. Image-to-image translation via group-wise deep whitening-and-coloring transformation. In *CVPR*, 2019.

Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In *CVPR*, 2018.

Dariusz Dereniowski and Marek Kubale. Cholesky factorization of matrices in parallel and ranking of graphs. In *International Conference on Parallel Processing and Applied Mathematics*, 2003.

Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur. A learned representation for artistic style. In *ICLR*, 2016.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *ICML*, 2015.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *JMLR*, 2016.

Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *CVPR*, 2016.

Boqing Gong, Yuan Shi, Fei Sha, and Kristen Grauman. Geodesic flow kernel for unsupervised domain adaptation. In *CVPR*, 2012.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014.

Gregory Griffin, Alex Holub, and Pietro Perona. Caltech-256 object category dataset. 2007.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *NIPS*, 2017.

Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei A Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation. In *ICML*, 2017.

Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. Multimodal unsupervised image-to-image translation. In *ECCV*, 2018.

Wei-Chih Hung, Varun Jampani, Sifei Liu, Pavlo Molchanov, Ming-Hsuan Yang, and Jan Kautz. Scops: Self-supervised co-part segmentation. In *CVPR*, 2019.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017.

Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *NIPS*, 2015.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv :1412.6980*, 2014.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.

Yanghao Li, Naiyan Wang, Jianping Shi, Jiaying Liu, and Xiaodi Hou. Revisiting batch normalization for practical domain adaptation. In *ICLR-WS*, 2017.

Ming-Yu Liu and Oncel Tuzel. Coupled generative adversarial networks. In *NIPS*, 2016.

Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. In *NIPS*, 2017.

Mingsheng Long and Jianmin Wang. Learning transferable features with deep adaptation networks. In *ICML*, 2015.

Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. Deep transfer learning with joint adaptation networks. In *ICML*, 2017.

Massimiliano Mancini, Lorenzo Porzi, Samuel Rota Bulò, Barbara Caputo, and Elisa Ricci. Boosting domain adaptation by discovering latent domains. In *CVPR*, 2018.

Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. Domain adaptation with multiple sources. In *NIPS*, 2009.

Takeru Miyato and Masanori Koyama. cgans with projection discriminator. In *ICLR*, 2018.

Pietro Morerio, Jacopo Cavazza, and Vittorio Murino. Minimal-entropy correlation alignment for unsupervised deep domain adaptation. In *ICLR*, 2018.

Zak Murez, Soheil Kolouri, David Kriegman, Ravi Ramamoorthi, and Kyungnam Kim. Image to image translation for domain adaptation. In *CVPR*, 2018.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS-WS*, 2011.

Xingchao Peng and Kate Saenko. Synthetic to real adaptation with generative correlation alignment networks. In *WACV*, 2018.

Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. *ICCV*, 2019.

Subhankar Roy, Aliaksandr Siarohin, Enver Sangineto, Samuel Rota Bulo, Nicu Sebe, and Elisa Ricci. Unsupervised domain adaptation using feature-whitening and consensus loss. *CVPR*, 2019.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, 2015.

Paolo Russo, Fabio Maria Carlucci, Tatiana Tommasi, and Barbara Caputo. From source to target and back: symmetric bi-directional adaptive gan. In *CVPR*, 2018.

Swami Sankaranarayanan, Yogesh Balaji, Carlos D Castillo, and Rama Chellappa. Generate to adapt: Aligning domains using generative adversarial networks. In *CVPR*, 2018.

Aliaksandr Siarohin, Enver Sangineto, and Nicu Sebe. Whitening and Coloring batch transform for GANs. In *ICLR*, 2019.

Baochen Sun and Kate Saenko. Deep coral: Correlation alignment for deep domain adaptation. In *ECCV*, 2016.

Baochen Sun, Jiashi Feng, and Kate Saenko. Return of frustratingly easy domain adaptation. In *AAAI*, 2016.

Antonio Torralba and Alexei A Efros. Unbiased look at dataset bias. In *CVPR*, 2011.

Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. Deep domain confusion: Maximizing for domain invariance. *arXiv:1412.3474*, 2014.

Eric Tzeng, Judy Hoffman, Trevor Darrell, and Kate Saenko. Adversarial discriminative domain adaptation. In *CVPR*, 2017.

Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv:1607.08022*, 2016.

Hemanth Venkateswara, Jose Eusebio, Shayok Chakraborty, and Sethuraman Panchanathan. Deep hashing network for unsupervised domain adaptation. In *CVPR*, 2017.

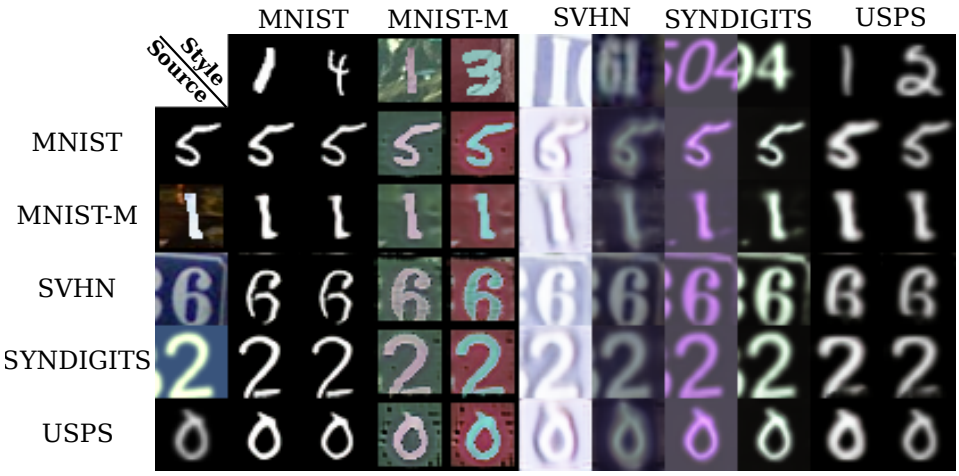
Ruijia Xu, Ziliang Chen, Wangmeng Zuo, Junjie Yan, and Liang Lin. Deep cocktail network: Multi-source unsupervised domain adaptation with category shift. In *CVPR*, 2018.

Yi Yao and Gianfranco Doretto. Boosting for transfer learning with multiple sources. In *CVPR*, 2010.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *CVPR*, 2017.

A ADDITIONAL MULTI-SOURCE RESULTS

Some sample translations of our $\mathcal{G}$ are shown in Fig. 3. For example, in Fig. 3 (a) when the SVHN digit "six" with side-digits is translated to MNIST-M the *cDWT* blocks re-projects it to MNIST-M domain (i.e., single digit without side-digits) and the *AdaIWT* block applies the instance-specific style of the digit "three" (i.e., blue digit with red background) to yield a blue "six" with red background. Similar trends are also observed in Fig. 3 (b).



(a) Sample translations of Digits-Five

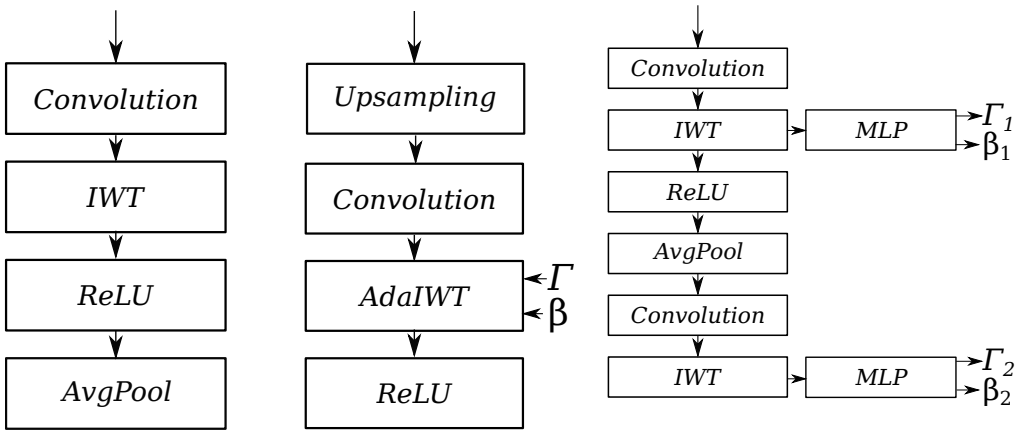
(b) Sample translations of Office-Caltech10

Figure 3: Generations of our $\mathcal{G}$ across different domains. Leftmost column shows the *source* images, one from each domain and the topmost row shows the *style* image from the target domain, two from each domain.

B IMPLEMENTATION DETAILS

In this section we provide the architecture details of the TriGAN generator $\mathcal{G}$ and the discriminator $\mathcal{D}_{\mathcal{P}}$.

Instance Whitening Transform (IWT) blocks. As shown in Fig 4 (a) each **IWT** block is a sequence composed of: $Convolution_{k\times k} - IWT - ReLU - AvgPool_{m\times m}$, where k and m denote the kernel sizes. There are two **IWT** blocks in the $\mathcal{E}$. In the first **IWT** block we use $k = 5$ and $m = 2$, and in the second we use $k = 3$ and $m = 2$.

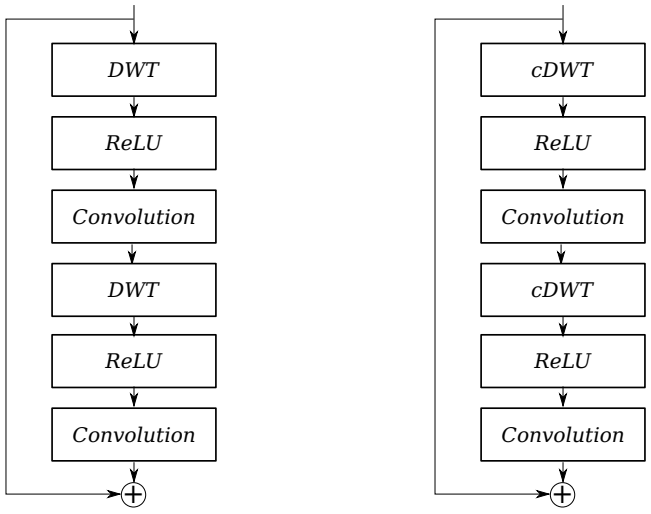


(a) **IWT** block (b) **AdaIWT** block (c) **Style Path**

Figure 4: A schematic representation of the (a) **IWT** block; (b) **AdaIWT** block; and (c) **Style Path**.

Adaptive Instance Whitening (**AdaIWT**) **blocks**. The **AdaIWT** blocks are analogous to the **IWT** blocks except from the *IWT* which is replaced by the *AdaIWT*. The **AdaIWT** block is a sequence: $Upsampling_{m\times m} - Convolution_{k\times k} - AdaIWT - ReLU$, where $m = 2$ and $k = 3$. *AdaIWT* also takes as input the coloring parameters ($\mathbf{\Gamma}$, $\boldsymbol{\beta}$) (See Sec. 3.2.3) and Fig. 4 (b)). Two **AdaIWT** blocks are consecutively used in $\mathcal{D}$. The last **AdaIWT** block is followed by a $Convolution_{5\times5}$ layer.

Style Path. The **Style Path** is composed of: $Convolution_{5\times5} - (IWT - MLP) - ReLU - AvgPool_{2\times2} - Convolution_{3\times3} - (IWT - MLP)$ (Fig. 4 (c)). The output of the **Style Path** is $(\boldsymbol{\beta}_1 \| \mathbf{\Gamma}_1)$ and $(\boldsymbol{\beta}_2 \| \mathbf{\Gamma}_2)$, which are input to the second and the first **AdaIWT** blocks, respectively (see Fig. 4 (b)). The MLP is composed of five fully-connected layers with 256, 128, 128, 256 neurons, with the last fully-connected layer having a number of neurons equal to the cardinality of the coloring parameters $(\boldsymbol{\beta} \| \mathbf{\Gamma})$.



(a) **DWT** block (b) **cDWT** block

Figure 5: Schematic representation of (a) **DWT** block; and (b) **cDWT** block.

Domain Whitening Transform (**DWT**) **blocks**. The schematic representation of a **DWT** block is shown in Fig. 5 (a). For the **DWT** blocks we adopt a residual-like structure He et al. (2016): $DWT - ReLU - Convolution_{3\times3} - DWT - ReLU - Convolution_{3\times3}$. We also add identity shortcuts in the **DWT** residual blocks to aid the training process.

Conditional Domain Whitening Transform (cDWT) blocks. The proposed **cDWT** blocks are schematically shown in Fig. 5 (b). Similarly to a **DWT** block, a **cDWT** block contains the following layers: $cDWT - ReLU - Convolution_{3\times3} - cDWT - ReLU - Convolution_{3\times3}$. Identity shortcuts are also used in the **cDWT** residual blocks.

All the above blocks are assembled to construct $\mathcal{G}$, as shown in Fig. 6. Specifically, $\mathcal{G}$ contains two **IWT** blocks, one **DWT** block, one **cDWT** block and two **AdaIWT** blocks. It also contains the **Style Path** and 2 $Convolution_{5\times5}$ (one before the first **IWT** block and another after the last **AdaIWT** block), which is omitted in Fig. 6 for the sake of clarity. $\{\mathbf{\Gamma}_1, \boldsymbol{\beta}_1, \mathbf{\Gamma}_2, \boldsymbol{\beta}_2\}$ are computed using the **Style Path**.

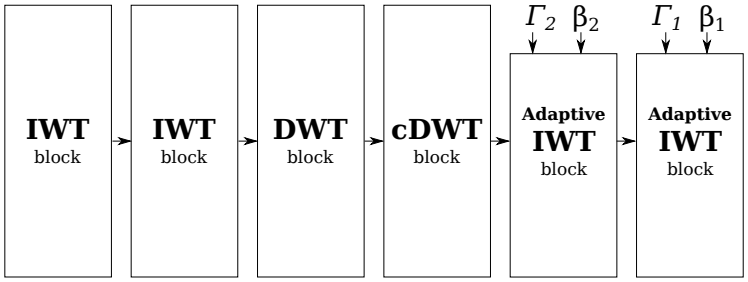


Figure 6: Schematic representation of the Generator $\mathcal{G}$ block.

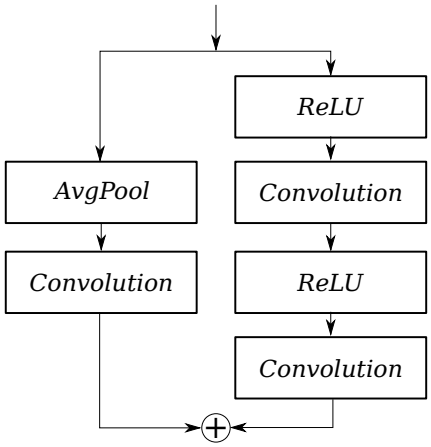


Figure 7: Schematic representation of the Discriminator $\mathcal{D}_\mathcal{P}$ block.

For the discriminator $\mathcal{D}_\mathcal{P}$ architecture we use a Projection Discriminator (Miyato & Koyama (2018)). In $\mathcal{D}_\mathcal{P}$ we use projection shortcuts instead of identity shortcuts. In Fig 7 we schematically show a discriminator block. $\mathcal{D}_\mathcal{P}$ is composed of 2 such blocks. We use spectral normalization (Miyato & Koyama (2018)) in $\mathcal{D}_\mathcal{P}$.

C Experiments for single-source UDA

Since, our proposed TriGAN has a generic framework and can handle N-way domain translations, we also conduct experiments for Single-Source UDA scenario where $N = 2$ and the source domain is grayscale MNIST. We consider the following UDA settings with the digits dataset:

C.1 Datasets

MNIST $\rightarrow$ **USPS**. The MNIST dataset contains grayscale images of handwritten digits 0 to 9. The pixel resolution of MNIST digits is 28×28. The USPS contains similar grayscale handwritten digits except the resolution is 16×16. We up-sample images from both domains to 32×32 during

training. For training TriGAN 50000 MNIST and 7438 USPS samples are used. For evaluation we used 1860 test samples from USPS.

MNIST → MNIST-M. MNIST-M is a coloured version of grayscale MNIST digits. MNIST-M has RGB images with resolution 28×28. For training TriGAN all 50000 training samples from MNIST and MNIST-M are used and the dedicated 10000 MNIST-M test samples are used for evaluation. Upsampling to 32×32 is also done during training.

MNIST → SVHN. SVHN is the short form of Street View House Number and contains real world version of digits ranging from 0 to 9. The images in SVHN are RGB with pixel resolution of 32×32. SVHN has non-centered digits with varying colour intensities. Presence of side-digits also makes adaption to SVHN a hard task. For training TriGAN 60000 MNIST and 73257 SVHN training samples are used. During evaluation all 26032 SVHN test samples are utilized.

Methods	Source Target	MNIST USPS	MNIST MNIST-M	MNIST SVHN
Source Only		78.9	63.6	26.0
DANN (Ganin et al. (2016))		85.1	77.4	35.7
CoGAN (Liu & Tuzel (2016))		91.2	62.0	-
ADDA (Tzeng et al. (2017))		89.4	-	-
PixelDA (Bousmalis et al. (2017))		95.9	<u>98.2</u>	-
UNIT (Liu et al. (2017))		95.9	-	-
SBADA-GAN (Russo et al. (2018))		<u>97.6</u>	**99.4**	<u>61.1</u>
GenToAdapt (Sankaranarayanan et al. (2018))		92.5	-	36.4
CyCADA (Hoffman et al. (2017))		94.8	-	-
I2I Adapt (Murez et al. (2018))		92.1	-	-
TriGAN (Ours)		**98.0**	95.7	**66.3**

Table 5: Classification Accuracy (%) of GAN-based methods on the Single-source UDA setting for Digits Recognition. The best number is in bold and the second best is underlined.

C.2 Comparison with GAN-based state-of-the-art methods

In this section we compare our proposed TriGAN with GAN-based state-of-the-art methods, both with adversarial learning based approaches and reconstruction-based approaches. Tab. 5 reports the performance of our TriGAN alongside the results obtained from the following baselines: Domain Adversarial Neural Network (Ganin et al. (2016)) (**DANN**), Coupled generative adversarial networks (Liu & Tuzel (2016)) (**CoGAN**), Adversarial discriminative domain adaptation (Tzeng et al. (2017)) (**ADDA**), Pixel-level domain adaptation (Bousmalis et al. (2017)) (**PixelDA**), Unsupervised image-to-image translation networks (Liu et al. (2017)) (**UNIT**), Symmetric bi-directional adaptive gan (Russo et al. (2018)) (**SBADA-GAN**), Generate to adapt (Sankaranarayanan et al. (2018)) (**GenToAdapt**), Cycle-consistent adversarial domain adaptation (Hoffman et al. (2017)) (**CyCADA**) and Image to image translation for domain adaptation (Murez et al. (2018)) (**I2I Adapt**). As can be seen from Tab. 5 TriGAN does better in two out of three adaptation settings. It is only worse in the MNIST → MNIST-M setting where it is the third best. It is to be noted that TriGAN does significantly well in MNIST → SVHN adaptation which is particularly considered as a hard setting. TriGAN is 5.2% better than the second best method SBADA-GAN for MNIST → SVHN.