**Noname manuscript No.**
(will be inserted by the editor)

# Supplementary Material. TriGAN: Image-to-Image Translation for Multi-Source Domain Adaptation

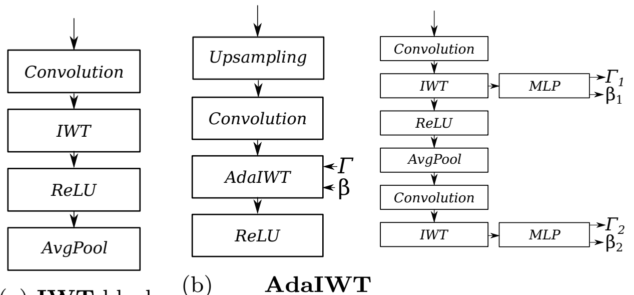
**Subhankar Roy · Aliaksandr Siarohin · Enver Sangineto · Nicu Sebe · Elisa Ricci**

Received: date / Accepted: date

## 1 Implementation details

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

**Instance Whitening Transform (IWT) blocks**. As shown in Fig 1 (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 $\mathcal{E}$. In the first **IWT** block, we use $k = 5$ and $m = 2$, while in the second we use $k = 3$ and $m = 2$.



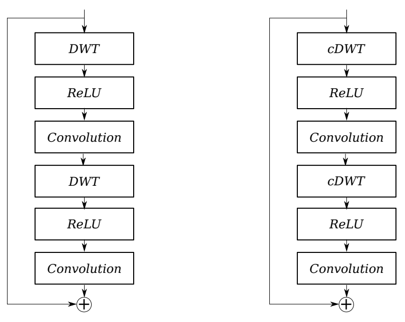
(a) **IWT** block (b) **AdaIWT** block (c) **Style Path**

Fig. 1: A schematic representation of (a) the **IWT** block; (b) the **AdaIWT** block; and (c) the **Style Path**.

**Adaptive Instance Whitening (AdaIWT) blocks**. The **AdaIWT** blocks are analogous to the **IWT** blocks, except from the *IWT* layers which are replaced with *AdaIWT* layers. Specifically, 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 ($\boldsymbol{\Gamma}$, $\boldsymbol{\beta}$) (see Sec. 3.2.3 of the main paper and Fig. 1 (b)). Two **AdaIWT** blocks are consecutively used in $\mathcal{D}$. The last **AdaIWT** block is followed by a $Convolution_{5\times 5}$ layer.

Subhankar Roy, Aliaksandr Siarohin, Enver Sangineto, Nicu Sebe and Elisa Ricci are with the Department of Information Engineering and Computer Science (DISI), University of Trento, Italy. Elisa Ricci is also with TeV, Fondazione Bruno Kessler (FBK), Trento 38050, Italy. E-mail: subhankar.roy@unitn.it

**Style Path**. The **Style Path** is composed of: *Convolution* $_{5\times 5} - (IWT - MLP) - ReLU - AvgPool_{2\times 2} - Convolution_{3\times 3} - (IWT - MLP)$ (Fig. 1 (c)). The output of the **Style Path** is $(\boldsymbol{\beta}_1 \| \boldsymbol{\Gamma}_1)$ and $(\boldsymbol{\beta}_2 \| \boldsymbol{\Gamma}_2)$, which are input to the second and the first **AdaIWT** blocks, respectively (see Fig. 1 (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 $(\boldsymbol{\beta} \| \boldsymbol{\Gamma})$.



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

Fig. 2: A schematic representation of (a) the **DWT** block; and (b) the **cDWT** block.

**Domain Whitening Transform (DWT) blocks**. The schematic representation of a **DWT** block is shown in Fig. 2 (a). For the **DWT** blocks we adopt a residual-like structure [3]: $DWT - ReLU - Convolution_{3\times 3} -$

$DWT - ReLU - Convolution_{3\times3}$. 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. 2 (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. 3. 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. 3 for the sake of clarity. $\{\boldsymbol{\Gamma}_1, \boldsymbol{\beta}_1, \boldsymbol{\Gamma}_2, \boldsymbol{\beta}_2\}$ are computed using the **Style Path**.

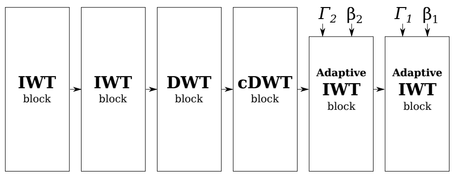


Fig. 3: A schematic representation of the Generator $\mathcal{G}$.

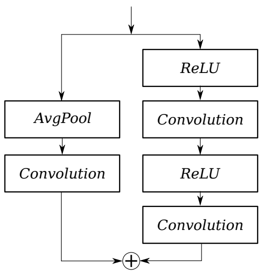


Fig. 4: A schematic representation of the Discriminator $\mathcal{D}_{\mathcal{P}}$.

For the discriminator $\mathcal{D}_{\mathcal{P}}$, we use a Projection Discriminator architecture [7]. In $\mathcal{D}_{\mathcal{P}}$ we use projection shortcuts instead of identity shortcuts. In Fig 4 we schematically show a discriminator block. $\mathcal{D}_{\mathcal{P}}$ is composed of 2 such blocks. We use spectral normalization [7] in $\mathcal{D}_{\mathcal{P}}$.

## 2 Experiments for single-source UDA

Since TriGAN has can handle $N$-source domain translations, we also conduct experiments for a Single-Source UDA scenario where $N = 1$ and the source domain is grayscale MNIST. Below we describe the adopted UDA settings with the Digits-Five dataset and the corresponding results.

### 2.1 Datasets

**MNIST → USPS**. The MNIST dataset contains grayscale images of handwritten digits from 0 to 9. The image resolution in MNIST is 28 × 28. USPS contains similar grayscale handwritten digits, except from the resolution which is 16 × 16. We up-sample the images of both domains to 32 × 32 during training. For training TriGAN, 50000 MNIST and 7438 USPS samples are used. For evaluation, we use 1860 test samples from USPS.

**MNIST → MNIST-M**. MNIST-M is a coloured version of the grayscale MNIST digits. MNIST-M has RGB images with resolution 28 × 28. For the TriGAN training, all the 50000 training samples from both MNIST and MNIST-M are used, and the dedicated 10000 MNIST-M test samples are used for evaluation. Training images are up-sampled to 32 × 32.

**MNIST → SVHN**. SVHN is the short form of Street View House Number and contains real-world images of digits, ranging from 0 to 9. The samples in SVHN are RGB images, with a resolution of 32 × 32. SVHN has non-centered digits with varying colour intensities. One challenging characteristic of the SVHN images is the presence of other digits, partially shown in the background. For the TriGAN training, 60000 MNIST and 73257 SVHN samples are used. During the evaluation, all the 26032 SVHN test samples are utilized.

### 2.2 Comparison with generation-based state-of-the-art methods

In this section we compare our proposed TriGAN with generation-based state-of-the-art UDA methods, either based on GANs or based reconstruction approaches. Tab. 1 reports the performance of our TriGAN alongside the results obtained from the following baselines: Domain Adversarial Neural Network [2] (**DANN**), Coupled generative adversarial networks [6] (**CoGAN**), Adversarial discriminative domain adaptation [11] (**ADDA**), Pixel-level domain adaptation [1] (**PixelDA**), Unsupervised image-to-image translation networks [5] (**UNIT**), Symmetric bi-directional adaptive gan [9] (**SBADA-GAN**), Generate to adapt [10] (**GenToAdapt**), Cycle-consistent adversarial domain adaptation [4] (**CyCADA**) and Image to image translation for domain adaptation

| Methods | Source<br>Target | MNIST<br>USPS | MNIST<br>MNIST-M | MNIST<br>SVHN |
|---|---|---|---|---|
| Source Only | | 78.9 | 63.6 | 26.0 |
| DANN [2] | | 85.1 | 77.4 | 35.7 |
| CoGAN [6] | | 91.2 | 62.0 | - |
| ADDA [11] | | 89.4 | - | - |
| PixelDA [1] | | 95.9 | <u>98.2</u> | - |
| UNIT [5] | | 95.9 | - | - |
| SBADA-GAN [9] | | <u>97.6</u> | **99.4** | <u>61.1</u> |
| GenToAdapt [10] | | 92.5 | - | 36.4 |
| CyCADA [4] | | 94.8 | - | - |
| I2I Adapt [8] | | 92.1 | - | - |
| TriGAN (Ours) | | **98.0** | 95.7 | **66.3** |

Table 1: Classification Accuracy (%) of generation-based methods on the *single-source* UDA scenario for digit recognition. The best value is in bold and the second best is underlined.

[8] (**I2I Adapt**). Tab. 1 shows that TriGAN outperforms all the other generative methods in two out of the three adaptation settings. In the MNIST $\rightarrow$ MNIST-M setting, TriGAN is the third best. It is interesting to note that TriGAN achieves significantly better results in the MNIST $\rightarrow$ SVHN setting, which is considered as a hard setting, where TriGAN is 5.2% better than the second best method SBADA-GAN.

## References

1. Bousmalis, K., Silberman, N., Dohan, D., Erhan, D., Krishnan, D.: Unsupervised pixel-level domain adaptation with generative adversarial networks. In: CVPR (2017)
2. Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., Lempitsky, V.: Domain-adversarial training of neural networks. JMLR (2016)
3. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)
4. 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)
5. Liu, M.Y., Breuel, T., Kautz, J.: Unsupervised image-to-image translation networks. In: NIPS (2017)
6. Liu, M.Y., Tuzel, O.: Coupled generative adversarial networks. In: NIPS (2016)
7. Miyato, T., Koyama, M.: cGANs with projection discriminator. In: ICLR (2018)
8. Murez, Z., Kolouri, S., Kriegman, D., Ramamoorthi, R., Kim, K.: Image to image translation for domain adaptation. In: CVPR (2018)
9. Russo, P., Carlucci, F.M., Tommasi, T., Caputo, B.: From source to target and back: symmetric bi-directional adaptive gan. In: CVPR (2018)
10. Sankaranarayanan, S., Balaji, Y., Castillo, C.D., Chellappa, R.: Generate to adapt: Aligning domains using generative adversarial networks. In: CVPR (2018)
11. Tzeng, E., Hoffman, J., Darrell, T., Saenko, K.: Adversarial discriminative domain adaptation. In: CVPR (2017)