UNDER REVIEW

RESCIENCE C

Replication / ML Reproducibility Challenge 2022

[Re] Exact Feature Distribution Matching for Arbitrary Style Transfer and Domain Generalization

Mert Erkol[1], Furkan Kınlı[1], Barış Özcan[1], and Furkan Kıraç[1]

[1]Ozyegin University, Vision and Graphics Laboratory, Istanbul, Turkey

Edited by
(Editor)

Received
01 November 2018

Published
–

DOI
–

Reproducibility Summary

In this reproducibility study, we present our results and experience during replicating the paper, titled Exact Feature Distribution Matching for Arbitrary Style Transfer and Domain Generalization [1]. In real-world scenarios, the feature distributions are mostly much more complicated than Gaussian, so only mean and standard deviation may not be fully representative to match them. This paper introduces a novel strategy to exactly match the histograms of image features via the Sort-Matching algorithm in a computationally feasible way. We were able to reproduce most of the results presented in the original paper both qualitatively and quantitatively.

Scope of Reproducibility – In the scope of this study, we aim to reproduce all the qualitative and quantitative results on two tasks, namely Arbitrary Style Transfer (AST) and Domain Generalization (DG). Moreover, we investigate the capability of forming better style representations by EFDM in another recent study [2].

Methodology – We have conducted all experiments in the original work by using the official repository, which is implemented by PyTorch [3]. For additional experiments, we have implemented the modular version of EFDM as a layer to replace it with the normalization modules. We have used 2 NVIDIA RTX 2080Ti GPUs for both training and testing, and it took roughly 1 day to complete a single training.

Results – We have reproduced the experiments done on two selected tasks, and compared their results with the reported results. Although our experimental results are not identical to the reported ones, we can validate the claims made by the original study according to these results.

What was easy – The paper is well-written and easy to follow. The original repository is well-organized to run all tests with the data presented in the paper.

What was difficult – The requirements in the repository were not updated, and we had to manage different versions of Python packages to be able to conduct the experiments.

Copyright © 2023 M. Erkol et al., released under a Creative Commons Attribution 4.0 International license.
Correspondence should be addressed to Mert Erkol (mert.erkol@ozu.edu.tr)
The authors have declared that no competing interests exist.
Code is available at https://github.com/birdortyedi/efdm-pytorch – DOI 10.5281/zenodo.7895753. – SWH swh:1:dir:b76a5bf3f3f540d17ef0f4a22ecc0b4e2c27d680;.
Open peer review is available at https://openreview.net/forum?id=a5_hbZf0NB&noteId=Rsj9NvSj2Ft.

Communication with original authors — We were in contact with the authors, and asked for the original results as JPEG files to prepare the figures in this report.

1 Introduction

Feature distribution matching is one of the most challenging learning tasks for visual inputs. Arbitrary Style Transfer (AST) and Domain Generalization (DG) are the common tasks in the literature where feature distribution matching can be considered as the solution. For example, in AST, the style information of input and target images can be interpreted as feature distributions and style can be transferred by cross-distribution feature matching [4, 5, 6, 7, 8, 9, 10]. The first drawback in the previous studies is to use only the mean and standard deviation to match the feature distributions, which mainly relies on the assumption of that the feature distributions follow Gaussian. In real-world scenarios, the feature distributions are often much more complicated than Gaussian, thus mean and standard deviation may not be fully representative to exactly match them. Secondly, although Exact Feature Distribution Matching (EFDM) can be achieved by directly matching the higher-order statistics of the image features, it is not practical for the current application areas due to the intensive computational overhead. This paper [1] proposes to perform EFDM in a more effective way by exactly matching the empirical Cumulative Distribution Functions (eCDFs) of image features. As mentioned in the paper, Glivenko–Cantelli theorem [11] states that the empirical Cumulative Distribution Function (eCDF) asymptotically converges to the Cumulative Distribution Function (CDF) when the number of samples approaches infinity. Relying on this theorem, this study demonstrates that the feature distributions (*i.e.*, mean, standard deviation, and higher-order statistics) can be exactly matched by using eCDFs. The authors claim that this can be achieved by employing a custom Exact Histogram Matching algorithm that implements Sort-Matching [12].
In this reproducibility report, we studied EFDM via the Sort-Matching algorithm on two tasks related to feature distribution matching. In this study, we aimed to reproduce the experiments provided in the original paper on AST and DG, and reported the details and issues we encountered during this process. We have compared the results obtained in our experiments with the ones reported in the original paper. We have also extended the experiments to observe how much the performance changes when some hyperparameters of EFDM or EFDMix are modified. In addition to the experiments in the paper, we have investigated the EFDM have the capability of forming better style representations in the cases of modeling the subjects as style.

2 Scope of reproducibility

The main idea of the paper is to introduce a novel strategy that achieves to exactly match the feature distributions by using eCDFs of the input and target image features. This strategy is tested on two tasks related to feature distribution matching, namely Arbitrary Style Transfer (AST) and Domain Generalization (DG).
The proposed EFDM strategy claims that it shows superior performance to the existing state-of-the-art methods of AST and DG in terms of visual quality and quantitative measures. To validate these claims and further analyze the proposed strategy, we try to investigate the following questions:

- Does EFDM work stably on AST and also more challenging photo-realistic style transfer scenarios?
- How can the style information of multiple images, which is extracted by EFDM, be interpolated in the feature space?
- Does the proposed feature augmentation method via EFDM (*i.e.*, EFDMix) improve the generalization capability on category classification and cross-domain instance retrieval?

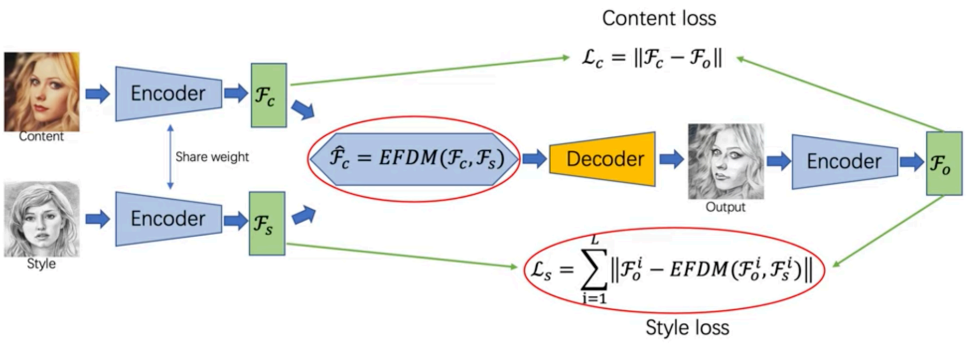

Figure 1. Overall training scheme of the proposed strategy in AST, including the EFDM of the image features. Obtained from the presentation of [1].

- Does the performance of the proposed strategy change by modifying the weight of the style loss term used in the training of EFDM and the instance-wise mixing weight used for EFDMix?
- Does EFDM have the capability of forming better style representations (*e.g.*, modeling the lighting as style [2])?

3 Methodology

The paper proposes a novel feature distribution matching strategy, namely EFDM via the Sort-Matching algorithm. We mainly focused on implementing the functionality of EFDM within the details described in the paper. For reproducing the results presented in the original paper, we have used the functional version of EFDM and the training pipelines of both AST and DG, as given in the official GitHub repository. The overall training scheme of AST and the usage of EFDM instead of the common normalization methods (*e.g.*, AdaIN [4]) can be seen in Figure 1. For our additional experiments on forming style representations by EFDM, we have implemented the modular version of EFDM as a layer to replace it with the common normalization modules.
We found that the paper is well-written and easy to follow. With the details given as supplementary material, the paper contains the important details required to reproduce all qualitative and quantitative results. However, the scripts provided in the official repository for t-SNE visualizations of higher-order statistics in the feature space does not work properly, and we could not achieve to fix it.
In this section, we introduce the implementation details of EFDM and further proposed feature augmentation strategy, namely EFDMix. We present the important points for the reproduction of this study, the hyperparameters we used, and our experimental setup.

3.1 Proposed Strategy

This study proposes to apply EFDM to tasks of AST and DG by exactly matching eCDFs with exact histogram matching via the Sort-Matching algorithm in feature space. Given the input vector $\mathbf{X} \in \mathbb{R}^{B \times C \times HW}$ and the style vector $\mathbf{Y} \in \mathbb{R}^{B \times C \times HW}$, EFDM can be applied by exact histogram matching in a channel-wise manner where B, C, H, W refer to the batch size, number of channels, height, and width, respectively. First, the values in $\mathbf{X}$ and $\mathbf{Y}$ are sorted in ascending order. To obtain the required output, the sorted values of $\mathbf{X}$ are replaced with the values of sorted $\mathbf{Y}$ in corresponding positions, then it returns the unsorted values of $\mathbf{X}$ whose elements are replaced with the values of $\mathbf{Y}$. In this way, the output will share the identical feature distribution to $\mathbf{Y}$. Note that it

Algorithm 1 PyTorch-like pseudo-code for EFDM.

```
X: input vector, Y: target vector
_, IndexX = torch.sort(X)
SortedY, _ = torch.sort(Y)
InverseIndex = IndexX.argsort(−1)
return X+ SortedY.gather(−1, InverseIndex) −X.detach()
```

requires applying the stop-gradient operation [13] to the style features, as practiced in the previous studies [4, 9], to ensure the flow of the gradients during back-propagation in deep models. The steps applied in practice are presented in Algorithm 1.
The proposed strategy does not introduce any additional parameters and can be used in a plug-and-play manner with few lines of code and minimal cost. It is important to note that the Sort-Matching algorithm assumes that two vectors (*i.e.*, $\mathbf{X}$ and $\mathbf{Y}$) should have the same number of dimensions in order to be directly applicable to this algorithm.
To extend this strategy for feature augmentation in DG, the authors introduce a style mixing method in feature space by interpolating the sorted vectors used in EFDM. This method is named as Exact Feature Distribution Mixing (EFDMix) in the paper. The main difference between EFDM and EFDMix is described in the following equation.

$$\mathbf{O} = \mathbf{X}_u + (1-\lambda)\mathbf{Y}_s - (1-\lambda)\mathbf{X}_d \tag{1}$$

where $\mathbf{O}$ stands for the required output, $\mathbf{X}_u$ is an unsorted input vector, $\mathbf{Y}_s$ is a sorted style vector, $\mathbf{X}_d$ refers to the gradient-stop operation applied to $\mathbf{X}_u$, and λ is the instance-wise mixing coefficient, which is sampled from $Beta(\alpha, \alpha)$ where $\alpha \in (0, \infty)$.

3.2 Architecture Design

A lightweight encoder-decoder architecture is employed for AST task where the encoder f is composed of the first 4 blocks of a pre-trained VGG-19 [14]. The decoder part is designed as a custom convolutional network that contains 4 convolutional blocks following ReLU activations [15]. Given the content images I_c and the style images I_s, both images are encoded into the feature space by using f. Note that the weights of f are fixed, and not trained during the experiments. EFDM is applied to these features in order to extract a new feature vector of the content images whose distribution is matched to the distribution of the style images. To summarize, the content features from the distribution of the style features S can be extracted by Equation 2.

$$S = EFDM(f(I_c), f(I_s)) \tag{2}$$

Decoder network g is responsible for projecting new stylized features S into the image space. The final output I_o can be generated by Equation 3.

$$I_o = g(S) \tag{3}$$

During the optimization of the weights of g, as a common practice in AST literature, the weighted combination of the content loss $\mathcal{L}_c$ and the style loss $\mathcal{L}_s$ is used, as shown in Equation 4.

$$\mathcal{L} = \mathcal{L}_c + \omega\mathcal{L}_s \tag{4}$$

Where ω is the balancing term for two components. The content loss refers to a simple Euclidean distance between the content images I_c and the final output I_o. The style loss calculates the distribution divergence between VGG features of the style images I_s and the final output I_o.

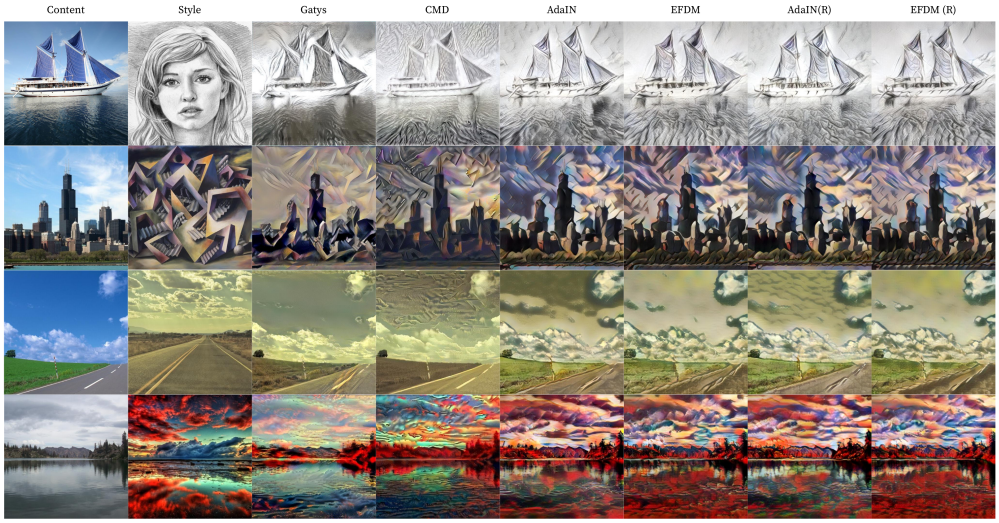

Figure 2. Visual comparison of the reported and our produced results on standard [4] (the first two rows) and photo-realistic [17] (the last two rows) style transfer.

For DG task, the original paper follows the prior work [9], and the only difference is to change the feature augmentation method used during training (*i.e.*, using EFDMix, instead of MixStyle [9]). ResNet-18 and ResNet-50 [16] are picked as a backbone network, and started to train these networks with pre-trained weights. There are two different settings in DG. The first is the leave-one-domain-out setting that trains the model on three domains and tests on the remaining one, and the latter is the single source generalization training on a single domain and testing on the remaining three domains.

3.3 Datasets

During our reproduction study, we used the same datasets and the same settings as mentioned in the original paper. The AST task is trained with the training set of MS-COCO [18] for the content images and WikiArt [19] for the style images. The training set of MS-COCO dataset contains 118K unique images, while WikiArt contains 42K images for training and 10K images for testing, collected from the artworks of 195 artists. For DG task, PACS dataset [20] is employed for domain generalization performance on image classification. This dataset contains images from four different domains (*i.e.*, Art Painting with 1.670 samples, Cartoon with 2.048 samples, Sketch with 2.344 samples, and Photo with 3.929 samples) with 7 shared categories. Moreover, Market1501 [21] and GRID [22] datasets are used for domain generalization on instance retrieval.

3.4 Hyperparameters

In our reproduction study, we used Adam optimizer [23] during training with an initial learning rate of $1e{-}4$, decay of $5e{-}5$, and the batch size of 8. The details for optimization are not available in the paper, and we have decided to use the default values as given in the official repository. In AST training, ω is set to 10 to adjust the content and style trade-off in the objective function. For DG task, α is the parameter for Beta distribution sampling, and is set to 0.1 during the experiments.

3.5 Experimental setup and code

We have followed the same protocol described in the original paper for both AST and DG. For any missing information in the paper, we abided by the default values given in the

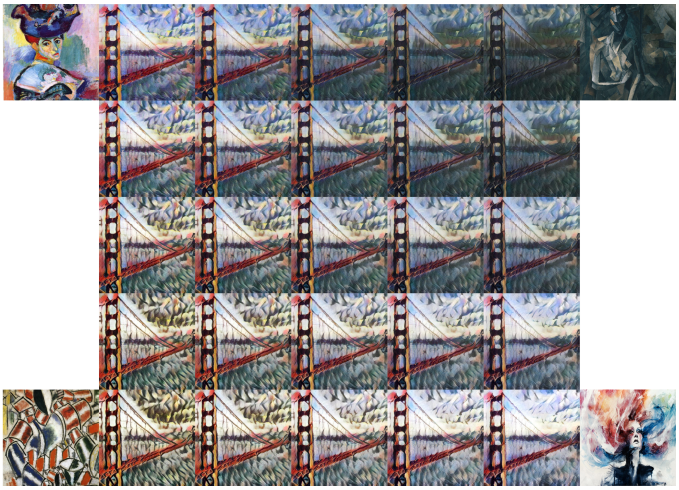

Figure 3. Illustration of style interpolation between a single content image and four style images.

Table 1. Average running time of prior methods and proposed strategy used in AST on a 512px image. Note that the compared methods run on a single Tesla V100, while our measurement has been done on a single RTX 2080Ti.

Method	Gatys *et al.* [24]	CMD [10]	HM	AdaIN [4]	EFDM [1]	EFDM (ours)
Time (s)	25.61	19.84	0.33	0.0038	0.0039	0.0043

official repository. We present a qualitative comparison to evaluate the performance of EFDM on AST. Following the original paper for DG task, the classification accuracy of the proposed strategy is reported in two specified settings (*i.e.*, leave-one-out generalization and single source generalization) and also the retrieval accuracy in the cross-dataset setting. For the additional experiments on modeling the lighting as the style, which is extracted by EFDM, we have followed the same training pipeline as introduced in [2], just replacing AdaIN layers with EFDM layers. Our implementation and the trained weights are available at the link[1].

3.6 Computational requirements

The experiments have been conducted on a single NVIDIA RTX 2080Ti GPU. A single training for AST task took approximately 12 hours, while all experiments for DG task has been completed in a single day. These experiments do not require any other significant resources, but GPU memory (*i.e.*, ~6GB for training of both tasks with the batch size of 8). The average running time of different methods used in AST to process a 512px image is shown in Table 1.

4 Results

We have conducted all experiments by following the descriptions given in the paper. In general, we were mostly able to reproduce the qualitative results on AST and photo-realistic style transfer, and quantitative results on DG. Reproduced results for both tasks support the claims made in the original paper. We can state that the overall performance seems robust to the changes in different hyper-parameters used for EFDM and EFDMix. Lastly, EFDM has the capacity to better represent the style information in the cases of modeling the lighting as style.

[1] https://anonymous.4open.science/r/efdm-pytorch-767F

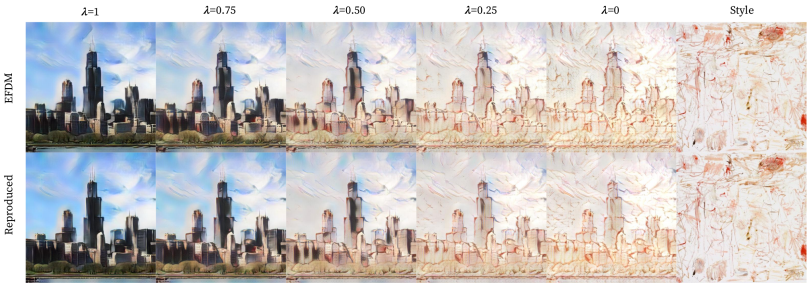

Figure 4. Illustration of the content-style trade-off with different λ values in Equation 1.

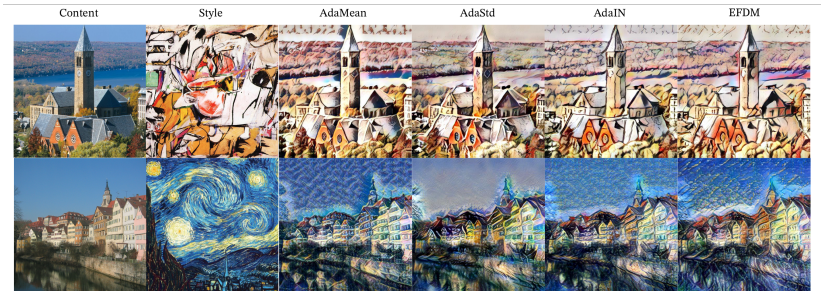

Figure 5. Comparison of reproduced results of different feature distribution matching strategies applied in the original paper.

4.1 Results reproducing original paper

Qualitative comparison on AST — As shown in Figure 2, we were able to reproduce the AST (the first two rows) and photo-realistic style transfer (the last two rows) results of AdaIN and EFDM reported in the original study. Although there could be minor differences in the corresponding outputs, depending on the optimization process, our reproduced models have similar behaviors on the same stylistic changes. Therefore, we can validate our first claim, EFDM works stably on AST and photo-realistic style transfer scenarios.

Mixing multiple styles — Figure 3 demonstrates the validation of our second claim. It is possible to blend more than one style information, instead of matching to a single one, to obtain novel styles by linearly combining their feature distributions.

Partial utilization of style information — The paper points out that the formula of EFDMix, given in Equation 1, enables adjusting the amount of style information utilized during style transfer. Figure 4 illustrates that we were able to reproduce the content-style trade-off experiment conducted in the original study.

EFDM versus different order of statistics — Following the ablation on AST in the original study, we present our reproduced results in Figure 5, where different feature distribution matching strategies are employed during AST training. AdaMean matches the dominant color scheme, while AdaStd tends to preserve the global structure more, instead of the stylistic details. AdaIN, by definition, can combine the behaviors of AdaMean and AdaStd. EFDM can effectively preserve the content details with the help of higher-order feature statistics.

Feature augmentation method via EFDM on DG — We present the domain generalization results of category classification in Table 2 and cross-domain instance retrieval in Table 3. We only report the results of the latest state-of-the-art [9], the original study [1], and our

Table 2. DG results of category classification on PACS. (R) refers to our reproduced results.

Method	Art	Cartoon	Photo	Sketch	Average
Leave-one-domain-out generalization					
R-18 w/ MixStyle [9]	83.1±0.8	78.6±0.9	95.9±0.4	74.2±2.7	82.9
R-18 w/ EFDMix [1]	83.9±0.4	79.4±0.7	96.8±0.4	75.0±0.7	83.9
R-18 w/ EFDMix (R)	80.6±1.5	78.1±0.6	94.1±0.9	72.3±1.2	81.3
R-18 w/ EFDMix (R) $\alpha = 0.5$	80.7±1.8	78.2±1.0	94.2±1.3	71.4±1.9	81.3
R-18 w/ EFDMix (R) $\alpha = 1.0$	80.9±1.4	78.1±0.9	94.1±1.3	71.4±2.1	81.1
R-50 w/ MixStyle [9]	90.3±0.3	82.3±0.7	97.7±0.4	74.7±0.7	86.2
R-50 w/ EFDMix [1]	90.6±0.3	82.5±0.7	98.1±0.2	76.4±1.2	86.9
R-50 w/ EFDMix (R)	87.4±1.6	81.8±1.6	94.3±2.2	73.7±1.7	84.3
R-50 w/ EFDMix (R) $\alpha = 0.5$	87.6±1.7	81.1±1.3	94.5±1.7	73.9±1.5	84.3
R-50 w/ EFDMix (R) $\alpha = 1.0$	87.4±2.1	81.6±1.4	94.7±1.6	74.3±1.6	84.5
Single source generalization					
R-18 w/ MixStyle [9]	61.9±2.2	71.5±0.8	41.2±1.8	32.2±4.1	51.7
R-18 w/ EFDMix [1]	63.2±2.3	73.9±0.7	42.5±1.8	38.1±3.7	54.4
R-18 w/ EFDMix (R)	63.5±3.4	72.9±1.2	41.9±1.4	36.3±3.1	53.7
R-18 w/ EFDMix (R) $\alpha = 0.5$	63.8±2.4	73.2±0.9	42.5±1.6	37.1±3.0	54.2
R-18 w/ EFDMix (R) $\alpha = 1.0$	63.7±3.4	73.2±0.9	41.9±1.8	36.3±2.4	53.7
R-50 w/ MixStyle [9]	73.2±1.1	74.8±1.1	46.0±2.0	40.6±2.0	58.6
R-50 w/ EFDMix [1]	75.3±0.9	77.4±0.8	48.0±0.9	44.2±2.4	61.2
R-50 w/ EFDMix (R)	73.0±2.2	77.2±0.9	48.3±1.2	47.7±2.7	61.6
R-50 w/ EFDMix (R) $\alpha = 0.5$	73.8±1.6	77.6±1.3	47.9±1.2	46.7±3.2	61.5
R-50 w/ EFDMix (R) $\alpha = 1.0$	73.7±1.2	77.8±0.4	47.9±0.7	46.0±4.2	61.4

Table 3. DG results on person re-ID task. (R) refers to our reproduced results.

Methods	MarKet1501 → GRID				GRID → MarKet1501			
	mAP	R1	R5	R10	mAP	R1	R5	R10
OSNet + MixStyle	33.8±0.9	24.89±1.6	43.7±2.0	53.1±1.6	4.9±0.2	15.4±1.2	28.4±1.3	35.7±0.9
OSNet + EFDMix	35.5±1.8	26.7±3.3	44.4±0.8	53.6±2.0	6.4±0.2	19.9±0.6	34.4±1.0	42.2±0.8
OSNet + EFDMix (R)	35.0±2.6	25.1±2.3	45.6±4.1	52.0±2.9	6.2±0.7	18.8±1.8	33.6±2.7	41.4±2.9

reproduction. We were able to reproduce the reported results of single source generalization experiments, while we could partially achieve to reproduce the reported results on the leave-one-domain-out generalization. Moreover, the original study claims that EFDMix outperforms the latest feature augmentation strategy for DG on cross-domain person re-identification, and our reproduced results can validate this claim.

4.2 Results beyond original paper

Trade-off between content and style loss terms — We investigate how much EFDM is robust to the weighting of two components in the objective function. As previously induced for AdaIN [4], the model inevitably starts to vanish the content details when the weight of style loss term is increased.

Modifying the instance-wise mixing coefficient — As shown in Table 2, the range parameters α of the distribution of the mixing coefficient in EFDMix does not have significant impact on DG results of category classification. This was expected since the method still mixes the distributions, and intuitively modifying its coefficient just makes it another distribution to be matched.

Forming better style representation — We further investigate the impact of using EFDM instead of AdaIN on a different domain. The approach [2] proposes to model the lighting as style to provide white-balance correction. This approach assumes that the illuminations in the scene basically stands for the additional style information injected to the scene,

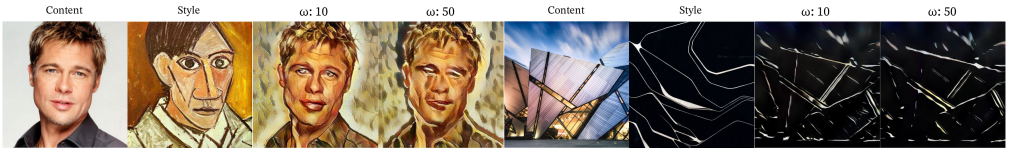

Figure 6. Ablation on the trade-off between content and style loss terms.

Table 4. White-balance correction results of the recent methods [2] and its variant with EFDM on mixed-illuminant evaluation set [25].

Method	MSE ↓				ΔE 2000↓			
	Mean	Q1	Q2	Q3	Mean	Q1	Q2	Q3
StyleWB [2]	822.77	572.52	840.67	1025.26	11.65	10.63	11.86	13.02
SR + AdaIN	818.99	527.34	875.56	1049.03	11.01	8.64	11.41	12.31
SR + EFDM	761.05	513.96	818.39	969.33	10.16	8.75	9.81	11.69

and tries to normalize this information in adaptive manner. In practice, we replaced AdaIN layers in this method with EFDM layers, which are implemented by us, and repeated the same experiment on mixed-illuminant evaluation set [25], as described in [2]. Table 4 demonstrates that EFDM forms better style representations to be utilized by proposed style removal model to remove the illumination.

5 Discussion

We can clearly say that the paper was well-written. Although there are some parts that we struggled in the official repository, we were able to run all necessary experiments requiring to reproduce this study. Overall, the reproduced results are similar to the reported results in the paper. As an exception, we could partially achieve to obtain comparable results on DG for category classification. In addition to this, we present the performance of the proposed method when some essential hyperparameters are slightly modified. Lastly, we extend the experiments to a different task in order to observe the impact of EFDM on forming style representations.

5.1 What was easy

The given code in the original repository was easy to follow, and it was well-written in general. The authors designed the documentation and the source code in a way that anyone who has fundamental knowledge of Python could run the experiments, or even generate their own stylized image from any content.

5.2 What was difficult

We would like to add the reproduced outputs by Histogram Matching (HM) along with the others, however the training of HM was based on CPU and the estimated time to complete a single training was around 15 days in our setup. Consequently, we could not include the reproduced outputs by HM to this report. Moreover, it could not be possible to add t-SNE visualizations to this report, as in the original paper, due to the lack of clarity in the documentation of its script.

5.3 Communication with original authors

We were in contact with the authors, and asked for the original results as JPEG files to prepare the figures in this report.

References

1. Y. Zhang, M. Li, R. Li, K. Jia, and L. Zhang. "Exact Feature Distribution Matching for Arbitrary Style Transfer and Domain Generalization." In: **CVPR**. 2022.
2. F. Kınlı, D. Yılmaz, B. Özcan, and F. Kıraç. "Modeling the Lighting in Scenes as Style for Auto White-Balance Correction." In: **Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision**. 2023, pp. 4903–4913.
3. A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. "Automatic differentiation in PyTorch." In: (2017).
4. X. Huang and S. Belongie. "Arbitrary style transfer in real-time with adaptive instance normalization." In: **Proceedings of the IEEE international conference on computer vision**. 2017, pp. 1501–1510.
5. Y. Li, N. Wang, J. Liu, and X. Hou. "Demystifying neural style transfer." In: **Proceedings of the 26th International Joint Conference on Artificial Intelligence**. 2017, pp. 2230–2236.
6. P. Li, L. Zhao, D. Xu, and D. Lu. "Optimal transport of deep feature for image style transfer." In: **Proceedings of the 2019 4th International Conference on Multimedia Systems and Signal Processing**. 2019, pp. 167–171.
7. M. Lu, H. Zhao, A. Yao, Y. Chen, F. Xu, and L. Zhang. "A closed-form solution to universal style transfer." In: **Proceedings of the IEEE/CVF International Conference on Computer Vision**. 2019, pp. 5952–5961.
8. Y. Mroueh. "Wasserstein Style Transfer." In: **International Conference on Artificial Intelligence and Statistics**. PMLR. 2020, pp. 842–852.
9. K. Zhou, Y. Yang, Y. Qiao, and T. Xiang. "Domain Generalization with MixStyle." In: **International Conference on Learning Representations**. 2020.
10. N. Kalischek, J. D. Wegner, and K. Schindler. "In the light of feature distributions: moment matching for Neural Style Transfer." In: (2021). arXiv:2103.07208 [cs.CV].
11. A. W. Van der Vaart. **Asymptotic statistics**. Vol. 3. Cambridge university press, 2000.
12. J. P. Rolland, V. Vo, B. Bloss, and C. K. Abbey. "Fast algorithms for histogram matching: Application to texture synthesis." In: **Journal of Electronic Imaging** 9.1 (2000), pp. 39–45.
13. X. Chen and K. He. "Exploring simple siamese representation learning." In: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition**. 2021, pp. 15750–15758.
14. K. Simonyan and A. Zisserman. "Very deep convolutional networks for large-scale image recognition." In: **arXiv preprint arXiv:1409.1556** (2014).
15. A. L. Maas, A. Y. Hannun, A. Y. Ng, et al. "Rectifier nonlinearities improve neural network acoustic models." In: **Proc. icml**. Vol. 30. 1. Atlanta, Georgia, USA. 2013, p. 3.
16. K. He, X. Zhang, S. Ren, and J. Sun. "Deep residual learning for image recognition." In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. 2016, pp. 770–778.
17. F. Luan, S. Paris, E. Shechtman, and K. Bala. "Deep photo style transfer." In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. 2017, pp. 4990–4998.
18. T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. "Microsoft coco: Common objects in context." In: **European conference on computer vision**. Springer. 2014, pp. 740–755.
19. W. R. Tan, C. S. Chan, H. Aguirre, and K. Tanaka. "Improved ArtGAN for Conditional Synthesis of Natural Image and Artwork." In: **IEEE Transactions on Image Processing** 28.1 (2019), pp. 394–409. DOI: 10.1109/TIP.2018.2866698. URL: https://doi.org/10.1109/TIP.2018.2866698.
20. D. Li, Y. Yang, Y.-Z. Song, and T. M. Hospedales. "Deeper, broader and artier domain generalization." In: **Proceedings of the IEEE international conference on computer vision**. 2017, pp. 5542–5550.
21. L. Zheng, L. Shen, L. Tian, S. Wang, J. Wang, and Q. Tian. "Scalable person re-identification: A benchmark." In: **Proceedings of the IEEE international conference on computer vision**. 2015, pp. 1116–1124.
22. C. C. Loy, T. Xiang, and S. Gong. "Multi-camera activity correlation analysis." In: **2009 IEEE Conference on Computer Vision and Pattern Recognition**. IEEE. 2009, pp. 1988–1995.
23. D. P. Kingma and J. Ba. "Adam: A method for stochastic optimization." In: **arXiv preprint arXiv:1412.6980** (2014).
24. L. A. Gatys, A. S. Ecker, and M. Bethge. "Image style transfer using convolutional neural networks." In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. 2016, pp. 2414–2423.
25. M. Afifi, M. A. Brubaker, and M. S. Brown. "Auto White-Balance Correction for Mixed-Illuminant Scenes." In: **Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)**. Jan. 2022, pp. 1210–1219.