Reproducibility report: Deep Fair Clustering for Visual Learning

Anonymous Author(s)
Affiliation
Address
email

Reproducibility Summary

Scope of Reproducibility

The authors propose a novel method for Deep Fair Clustering (DFC), combining existing frameworks for fair clustering—which typically have difficulty with high-dimensional large-scale data—with previous work on deep clustering—which typically has difficulty with fairness. Our reproducibility work targets the central claim that DFC learns fair representations with minimal utility loss and obtains superior results on both fairness and accuracy.

Methodology

We used the code repository made available by the authors and extended it to include support for pretraining, different datasets and comparative methods. We compare the DFC method against Deep Embedded Clustering (DEC) (11), —which implements a comparable deep clustering method without fairness constraints—on the same four datasets (obtained from MNIST (7), USPS (6), MTFL (13) and Office-31 (9)) and fairness metrics that were used in the paper. We select one dataset for additional experiments aimed at validating the contribution of individual components of the DFC towards fairness. All experiments were run on a GeForce 1080Ti GPU. Hyperparameter optimization was performed using the Weights & Biases Sweeps feature (1).

Results

On the selected dataset, we reproduced accuracy to within 2% of reported value, normalized mutual information (NMI) and Entropy to within 1%, and balance to within 5%. Our DFC method outperformed our DEC method on all accuracy and fairness metrics. We reproduced the accuracy of the non-digit datasets to within 1% (Office-31) and 5% (MTFL) but failed to obtain similar results for balance.

What was easy

We found no major challenges reproducing the provided code in as far as we used their provided pretrained models and the selected dataset used in the code.

What was difficult

Extending the code for the non-digit datasets was a challenge, as some hyperparameter settings and architecture details were difficult to infer from the paper. We found that performance was sensitive to small changes in implementation and training of the encoders, causing us to run into time and resource constraints when trying to reproduce all results for these datasets due to the large number of models that require pretraining.

Communication with original authors

We had helpful one-off contact with the authors to verify hyperparameter settings.

Submitted to ML Reproducibility Challenge 2020. Do not distribute.

1 Introduction

Machine learning (ML) is increasingly used in high-stake decision-making where the data might contain sensitive attributes, such as gender, race or socioeconomic background. Examples include college admission, loan approval and bail/parole judgements. Such ML algorithms are vulnerable to bias and unfairness (4), and extensive literature has brought attention to the various challenges and inherent trade-offs in this field (5)(14).

Clustering is an important aspect of many such applications. To illustrate what it might mean for a clustering assignment to be unfair, we can consider its use in feature engineering, e.g., to label data in an unsupervised setting to increase its expressive power (2). In some cases, if the inherent structure of the training data differs between subgroups, standard clustering is likely to lead to partition assignments that correlate significantly with sensitive attributes. This could facilitate subsequent discrimination based on sensitive attributes, either knowingly or unknowingly.

In short, clustering is an active and complex field that still faces many challenges. The paper 'Deep Fair Clustering for Visual Learning' addresses various such challenges. Moreover, their method for fair clustering outperforms competitive fair clustering methods as well as competitive deep clustering methods, showing that their method is able to impose fairness constraints without significantly compromising clustering accuracy.

2 Scope of reproducibility

The paper aims to tackle the problem of fair clustering of high-dimensional data by introducing a Deep Fair Clustering (DFC) method suitable for image data. Their notion of fairness extends the demographic equipartition criterion and states that the clustering assignment should be independent of the protected subgroup membership. To achieve this independence, they leverage the feature representation encoding from deep methods and train the model to filter out sensitive attributes from the representations. Simultaneously, DFC enforces fairness constraints and optimizes clustering performance, notably with minimal trade-off between the two. The central claims are summed up as follows:

- The first claim states that DFC achieves fair clustering of large-scale and high-dimensional visual data by learning fair representations of the input. To support this claim, we train DFC and DEC on the MNIST-USPS dataset and compare the learned representations in figure 2. The first row of table 1 shows the corresponding accuracy and fairness metrics of the final performance of both methods.
- The second claim regards the validity and effectiveness of the proposed minimax optimization formulation. In particular, we support this claim by recognizing DEC as DFC without the minimax optimization formulation (in particular, without the fairness components 4, 5 and the separate clustering of subgroups, see figure 1). To our best knowledge, this corresponds to the implementations in (11) and (8). The claim is supported by comparing DFC performance with DEC performance on accuracy and fairness metrics in table 1.
- The third claim states that DFC shows superior performance on four real-world visual datasets (see 3.1). We aim to show this by running experiments on the same four datasets with both DEC and DFC and compare performance in table 1.

3 Methodology

Each claim requires us to implement the DFC method and the DEC method and train it on the MNIST-USPS dataset. From here, we address each claim separately in three sets of experiments. For the first, we visualize the representations using dimensionality reduction methods and compare the obtained clusters. For the second claim, we compare the DEC and DFC on accuracy, NMI, Balance and Entropy metrics. For the third claim, we run our experiments with the three other datasets that are used in the paper and compare results on the above mentioned four metrics.

Before moving on to the details we briefly explain the DFC method as we implemented it based on the authors description. The method has several components that are trained simultaneously in a minimax optimization scheme. A full overview of the method is shown in figure 1. The task is to cluster datapoints X with sensitive categorical attribute G belonging to protected subgroup $G(X) \in [M]$ into K clusters. A feature encoder $\mathcal{F}(X)$ transforms the data X into representations Z. As the figure shows, each subgroup is clustered separately and requires a separate, pretrained encoder. The clustering assignment $P = \mathcal{A}(Z)$, taken from the Deep Embedded Clustering method (DEC), creates a soft assignment P reflecting the probability of datapoints to be assigned to each cluster, once for each protected subgroup, and once for the full dataset.

A discriminator $\mathcal{D}$ aims to reconstruct the protected subgroup membership based on the soft assignment P

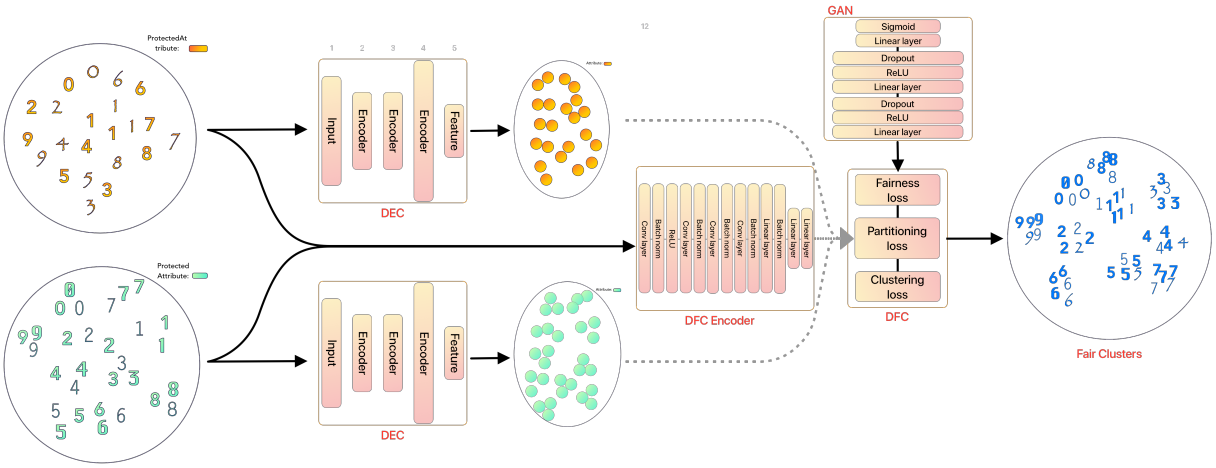

Figure 1: Overview of Deep Fair Clustering. The orange and green colors represent the protected subgroups.

Encoder Central to the DFC approach is the encoding of fair representations. Transforming the input data using feature encoding is important, in the first place, because (deep) clustering at pixel level does not often yield good results. However, in DFC the encoded representations are at the same time an important step towards fairness. The authors note that the representations can be filtered from the sensitive attributes, noting that "fair representations are achieved when the discriminator cannot distinguish between representations from different protected subgroups." To this end, the encoder is finetuned simultaneously with the rest of the clustering algorithm.

Discriminator A discriminator is implemented to provide a measure of fairness. It is trained to discriminate protected subgroup membership based on the soft assignment P.

Deep Embedded Clustering For the clustering assignment, the authors closely follow the DEC method. As in (11), the optimal cluster assignment is learned using Stochastic Gradient Descent (SGD),repeating the following two processes:

1. A soft assignement of the datapoints to cluster centroid using the Student's t-distribution as a kernel to measure the similarity between embedded point z and centroid c_k:

$$p_k = \frac{(1 + ||z - c_k||^2/\alpha)^{-\frac{\alpha+1}{2}}}{\sum_{k'}(1 + ||z - c_{k'}||^2/\alpha)^{-\frac{\alpha+1}{2}}}, \tag{1}$$

   Where p_k is the probability of a datapoint belonging to cluster center c_k and α the degrees of freedom of the Student's t-distribution.

2. The obtained distribution P of soft assignments is matched to an auxiliary distribution Q:

$$q_k = \frac{(p_k)^2 / \sum_{x \in X_g} p_k}{\sum_{k' \in [K]}((p_{k'})^2 / \sum_{x \in X_g} p_{k'})}, \tag{2}$$

   which is calculated for each protected subgroup separately. Finally, the clustering is optimized by minimizing the KL-divergence loss:

$$\mathcal{L}_c := KL(P||Q) = \sum_{g \in [M]} \sum_{x \in X_g} \sum_{k \in [K]} p_k \log \frac{p_k}{q_k}, \tag{3}$$

   Which ensures the soft assignment matches the target distribution as closely as possible.

Note that p_k, z and c_k are optimized simultaneously, such that both the P and Q distribution change dynamically.

Deep Fair clustering DFC implements DEC as above, but adds two loss function to impose the fairness constraint while promoting cluster utility. To encourage fair partitions according to the demographic parity, the authors introduce an *fairness adversarial loss*:

$$\mathcal{L}_f := \mathcal{L}(\mathcal{D} \circ \mathcal{A} \circ \mathcal{F}(X), G), \tag{4}$$

Where $\mathcal{L}$ is the cross-entropy loss function and $\circ$ denotes function composition, in particular the result of the encoder $\mathcal{F}(X)$, the clustering assignment $\mathcal{A}$ and the discriminator $\mathcal{D}$ applied to the data in sequence. The discriminator $\mathcal{D}$

is implemented as a Generative Adversarial Network (GAN) (see figure 1) with the exact inverse architecture of the used encoder. It monitors how well the DFC is able to reconstruct the protected subgroup based on the predicted cluster probabilities of a datapoint. Effectively, with respect to the *fairness adversarial loss*, $\mathcal{D}$ is maximized with the probability of assigning the correct membership label for each sample. Simultaneously, clustering $\mathcal{A}$ and representations $\mathcal{F}(X)$ are trained to maximally confuse $\mathcal{D}$.

To increase the clustering performance under the fairness constraint, DFC adds a *structure preservation loss*:

$$\mathcal{L}_s = \sum_{g \in [M]} \left|\left| \hat{P}_g \hat{P}_g^\top - P_g P_g^\top \right|\right|^2, \tag{5}$$

Where $\hat{P}_g$ is the soft assignment for a datapoint in protected subgroup g, and P_g is the result for subgroup g when the clustering was done over the complete dataset. In other words, they expect, given a subgroup, that the clustering result is similar when trained only on the subgroup data and when trained on all subgroups and looking only at the samples of one subgroup, and encourage the distributions to be similar.

The final learning objective is as follows:

$$\max_{\mathcal{F},\mathcal{A}} \quad \alpha_f \mathcal{L}_f - \alpha_s \mathcal{L}_s - \mathcal{L}_c \tag{6}$$

$$\min_{D} \quad \alpha_f \mathcal{L}_f \tag{7}$$

Note how the fairness adversarial loss (4) is simultaneously maximized w.r.t. $\mathcal{F}$ and $\mathcal{A}$ (6) to minimize performance of the discriminator as a result of clustering, and minimized w.r.t $\mathcal{D}$ (7) to optimize its parameters and obtain a saddle-point solution. The values for α are 1 by default. Following the paper, for the office-31 dataset α_s is recalculated as $(512/128)^2 \cdot (31/10)$ to adjust for difference in batch size and cluster number.

3.1 Datasets

Following the original paper, the following four datasets were used in our experiments:

MNIST-USPS is obtained from combining the MNIST dataset (7)[1] and the USPS dataset (6) [2]. From both, only the train images are taken (67,291 images in total, 60,000 and 7,261 respectively). Both datasets consist of images of handwritten digits, with a dimensionality of 1024 (32x32 pixels). The number of clusters is 10, corresponding to the number of single digits. The sensitive attribute is the source of an image (MNSIT or USPS). The clusters are well balanced in both datasets (2.301 and 2.270 for MNIST and USPS respectively) but as USPS is much smaller, the maximum balance is 0.120.

Color Reverse MNIST is obtained by reversing the black and white pixels in the MNIST dataset and combining the normal and the reversed version (120,000 images in total). Here, the sensitive attribute is whether the image comes from the normal or the reversed version. The entropy is 2.303 for both subgroups and the balance is 1.00

The *The Multi-task Facial Landamark (MTFL)* dataset (13) constist of 12,995 face images annotated with gender, smiling, with or without glasses and the degree of the pose. The authors set the glasses as the sensitive attribute and cluster for gender. For balanced subgroups, 1000 images with and 1000 without glasses are sampled randomly, yielding a perfect balance of 1.00. The (binary) entropy is 0.693.

The Office-31 dataset (9) consists of images from 31 different categories collected from two distinct domains: Amazon and Webcam. Each domain contains all the categories. Amazon and Webcam are selected for the experiments, because these two have the largest domain divergence, and the domain source is set as the protected attribute. As the Amazon domain contains more data samples, the maximum balance is 0.273. The maximum entropy is 3.434.

3.1.1 Hyperparameters

The DFC model is dependent on 5 pretrained models; a VAE and a DEC for each subgroup and a VAE for the whole dataset. Since the optimal hyperparameters of the DFC are likely to depend on the hyperparameter choices of the pretrained models, a complete search would be very expensive. To make hyperparameter search manageable the assumption is made that the optimal hyperparameters are independent of the input data and the pretrained models it uses. This assumption enables to search the hyperparameter space of the DFC, DEC and VAE independently. Note

[1] http://yann.lecun.com/exdb/mnist/

[2] https://www.kaggle.com/bistaumanga/usps-dataset

that the VAE only needs to be pretrained and optimized for the MNIST-USPS and Reverse MNIST datasets, the other datasets use a pretrained ResNET50 encoder.

The hyperparameter search for the DEC and the VAE were focused on finding the best combination of the learning rate, batch size and the number of epochs. These searches were performed with Bayesian optimization and contained respectively 32 and 18 experiments on MNIST. The VAE Bayesian search was set to maximize the accuracy of the KMeans clustering afterwards, which is also used in the training of the DFC as initial centroids. The goal of the DEC was to optimize its clustering accuracy.

The DEC and VAE were pretrained using the optimal parameters on MNIST and USPS to facilitate the hyperparameter search of the DFC. The focus was on finding the optimal learning rate, batch size, and number of epochs. A random grid search was used since it is not clear which metric should be optimized. The best performing set of parameters is chosen based on the Accuracy, NMI, entropy and balance.

The range of the hyperparameter space was inspired by hyperparameters mentioned by the authors in the paper, email contact and a DEC MNIST example[3].

The hyperparameters of the KMeans clustering were set to the same values the authors used. TSNE hyperparameters were set to defaults as can be seen in Appendix C, which contains a complete list of all hyperparameters and overviews of the hyperparameter searches.

4 Experimental setup and implementation

All experiments were run on cuda enabled computers with seeds set to 2019 before training each model, unless otherwise specified. Our code containing the experiments is available at our GitHub repo [4]. Our code is strongly based on the implementation provided by the authors [5]. We have found that their code is a clear reflection of the paper. However, it was necessary to extend their implementation to provide support for pretraining, other datasets and other methods.

The DFC architectures described in the paper relies on several models joint together. Roughly, it includes an encoder, a clustering assignment and a discriminator. Some experiments require sightly different model combinations, which is always made explicit.

4.1 Pretrained Encoder

The type of encoder can be chosen independently from the DFC. The authors use different architectures based on the complexity of the dataset; for the digit based datasets a standard Variational Autoencoder is used, for the other datasets a pretrained ResNet50.

The standard VAE is based on four blocks of a 2D convolutional layer, batch normalization and a ReLU activation layer. This is scaled down in 2 steps with fully connected layers to the 64 dimensional hidden space. Since the decoder was missing from the author's implementation, some guesses had to be made about the exact implementation, we tried to follow conventions as much as possible. A standard KL-Divergence and MSE loss were used for pretraining. The exact architecture can be found in Appendix A.

For MTFL and Office-31, considering their complex and high resolution images, the authors use ImageNet-pretrained ResNet50 (3). The authors did not mention how the ResNet is finetuned exactly. Due to the relatively small size of the datasets used for the experiments, it was decided to only finetune the last layer during training the DFC.

4.2 Pretrained DEC

The structural preservation loss, discussed before 5, is calculated using two separate DEC models. These models could be seen as the golden standard for each subgroup. Since they are equally weighted, this ensures that the protected subgroup is underrepresented in the loss. These DECs also require an encoder to transform the images. The same encoder training methods and architectures are used for all encoders. The DECs were implemented as a stripped down DFC with three modifications. The fairness and the structural preservation loss were removed. Lastly, the clustering loss is calculated over the entire batch at once, instead of individually per subgroup. This results in an exact implementation of the original DEC paper (11).

[3] https://github.com/vlukiyanov/pt-dec/blob/master/examples/mnist/mnist.py

[4] https://github.com/Joppewouts/Reproducing-Deep-Fair-Clustering

[5] https://github.com/brandeis-machine-learning/DeepFairClustering

4.3 Centroid Initialization

The deep clustering methods DEC and DFC both benefit from properly initializing the Student's t-dsitribution cluster definitions. This is done with a simple KMeans clustering approach. KMeans is trained on the entire dataset that is used by the clustering method afterwards. The final centroids learned are copied to be the initial cluster definitions.

4.4 DFC

The DFC consists of a pretrained encoder, a discrminator module and cluster assignment module. The weights of these modules are updated with the Adam optimizer, using an inverse exponential learning rate scheduler. The discriminator has a standard architecture, details are attached in Appendix B. As described by the authors, the learning rate of the discriminator uses a multiplication factor of 10.

4.4.1 Clustering Assignment

The fairness constraints and learning objectives of DFC rely on clustering being performed on the subgroups separately. Since these subgroups are homogeneous with respect to sensitive attributes, they can be clustered irrespective of fairness-considerations. Therefore, the same model can be use as in the Deep Embedding clustering. Accordingly, the authors use the DEC clustering for the subgroups separately, which consists of a method similar to iterative k-means clustering, where a Student's t-distribution is used to measure similarity between an embedded representation and a cluster centroid, followed by KL-divergence minimization. For full details see (11).

4.5 Competitive methods

Due to time constraints the decision was made to only implement one competitive method. DEC was chosen since this can be seen as the direct ancestor model of the DFC. As explained before, this method was implemented as a stripped down version of the DFC which enables the evaluation on individual subgroups while training on the complete set.

4.6 Metrics

Four metrics are used to evaluate performance; accuracy, normalized mutual information, balance and entropy. These are the same metrics as in the original paper, which includes detailed explanations. Furthermore, for evaluating the fairness two more metrics are implemented. The accuracy of the discriminator is computed to function as a proxy for the fairness of the cluster assignments. The probabilities outputted by the discriminator are converted to subgroup classes by thresholding at 0.5. An accuracy of 50% would indicate that the discriminator cannot guess the subgroup class of a sample by evaluating the cluster assignment, and therefore fair cluster assignments. Secondly, TSNE (10) is implemented to visualize the latent space learned by the encoder. In an ideal fair representation, the representation of the two subgroups should be indistinguishable. For example, the digit 5 of MNIST and the digit 5 of USPS both encoded as the character '5' would be ideal, since the representations are indistinguishable. The goal of a TSNE mapping is that distant samples in the high dimensional space are also far away from each other in the lower dimension, and vice versa. Therefore, in a visualization of a fair latent space, it should be hard to distinguish clusters of protected subgroups. All results are calculated over 5 different seeds: 0, 1, 42, 2019 and 2020. However, it was not possible to provide the variance for the results that require a ResNet encoder due to time and resource limitations.

4.6.1 Computational requirements

We trained all models on an 8.25MB GeForce 1080Ti GPU. Pretraining VAE encoders for each dataset took $(5\times) \pm 45$min, pretraining the DEC models $(7\times) \pm 1$hr. The sweep for optimal parameter search took $(20\times)$ 1hr and running the DFC and DEC models for all four datasets on 5 random seeds took $(8\times) \pm 1$hr GPU time. The total experiments required a total of ± 40hr GPU time.

5 Results

Our results support the first claim that DFC learns fair representations that do not cause much loss in utility and the second claim that the minimax optimization formulation contributes to clustering that is both fair and accurate. We reproduced accuracy on the MNIST-USPS dataset to within 2% of reported value, normalized mutual information (NMI) and Entropy to within 1%, and balance to within 5%. Our DFC method outperformed our DEC method on all accuracy and fairness metrics. We reproduced the accuracy of the Office-31 datasets to within 1% and the MTFL dataset to within 5%, but failed to obtain similarly good results for the fairness metrics.

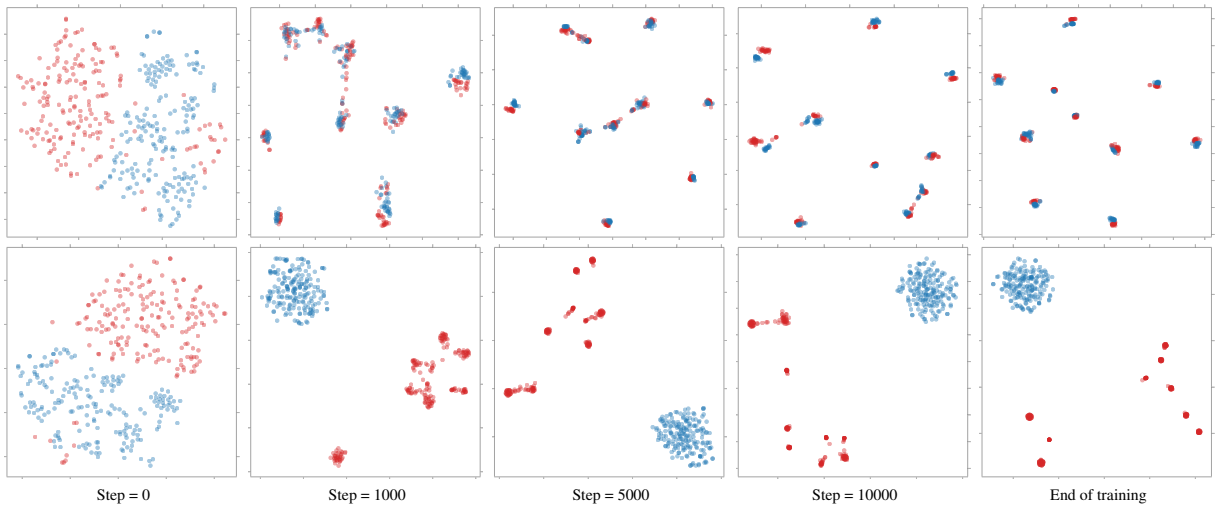

Figure 2: TSNE visualizations of the latent space learned by the encoder on MNIST-USPS with the final models. The rows respectively show the representations for the DFC and for the DEC. The DFC learns representations of the images which results nicely in 10 clusters in the visualized space. The DEC learns 9 distinct representation for MNIST digits and does not seem to separate USPS digits at all.

5.1 Claim 1: Learning fair representations for the MNIST-USPS dataset

Figure 2 shows the learned representation of the DFC method (top) and the DEC method (bottom). Protected subgroup membership is indicated by color for MNIST (red) and USPS (blue). Monitoring the DFC discriminator performance shows that the accuracy for recovering the protected subgroup drops from 60% at the first 1000 runs to 50% after 2000 iterations. The corresponding accuracy and fairness metrics are shown in table 1. We obtain an accuracy within 2% of the accuracy reported in the paper. NMI and Entropy scores fell within 1% of the original findings, and balance was 5.5% lower in our experiments.

5.2 Claim 2: Validity of minimax optimization formalization

The first two rows of table 1 show performance on all four metrics for the DFC and DEC methods. Results indicate that DFC outperforms DEC both in accuracy and in fairness methods. Moreover, the table shows that our implementation of the DEC method achieves a 17% higher accuracy and performs better than the original implementation of the DEC on all other metrics except balance.

5.3 Claim 3: Different datasets

The results for all datasets are can be found in table 1. Our implementation of DFC outperformed DEC on the three other datasets and showed similar accuracy as reported in the original paper. Results for entropy were also consistently higher than the DEC implementations and similar to the entropy reported in the original paper. The balance, however, was much lower than in the original paper, and did not differ significantly from the DEC results.

6 Discussion

Our results show that the reported performance of DFC can be reproduced on the MNIST-USPS digit dataset. We were partly unable to reproduce its fairness performance on the three other datasets but can think of various limitations of our approach that could provide an explanation for this. First, we put notable effort into adjusting and training pretrained models for the MNIST-USPS experiments, and performed a hyperparameter search to find optimal settings. Due to time and resource constraints, we were not able to repeat this process for all datasets, which may have led to different results. Secondly, our use of the ResNet50 encoder differed slightly from that in the paper. Correspondence with the authors had confirmed that their implementation finetuned all parameters of ResNet50. However, for us GPU memory constraints made it impossible to store all gradients that would be required. Therefore, we resolved to only finetune the final layer, based on transfer learning literature (12) and the relatively small sizes of the datasets. This could partially explain our different results.

Dataset		Method	Acc		NMI		Balance		Entropy	
MNIST-USPS	ours	DFC	0.81	(0.012)	0.80	(0.015)	0.012	(0.001)	2.292 / 2.299	(0.004 / 0.001)
		DEC	0.76	(0.004)	0.73	(0.003)	0.000	(0.000)	2.166 / 2.278	(0.007 / 0.002)
	orig.	DFC	0.83		0.79		0.067		2.301 / 2.265	
		DEC	0.59		0.69		0.000		2.082 / 1.735	
Reverse MNIST	ours	DFC	0.39	(0.027)	0.37	(0.038)	0.007	(0.001)	2.283 / 2.293	(0.009 / 0.002)
		DEC	0.32	(0.010)	0.32	(0.018)	0.000	(0.000)	1.395 / 1.939	(0.118 / 0.055)
	orig.	DFC	0.58		0.68		0.763		2.294 / 2.301	
		DEC	0.40		0.48		0.000		1.774 / 1.384	
MTFL	ours	DFC	0.79		0.27		0.001		0.655 / 0.693	
		DEC	0.77		0.23		0.001		0.305 / 0.693	
	orig.	DFC	0.72		0.19		0.986		0.693 / 0.693	
		DEC	0.52		0.03		0.711		0.660 / 0.576	
Office 31	ours	DFC	0.68		0.72		0.000		2.689 / 3.393	
		DEC	0.59		0.65		0.000		2.300 / 3.079	
	orig.	DFC	0.69		0.72		0.196		3.422 / 3.403	
		DEC	0.55		0.60		0.000		3.063 / 2.937	

Table 1: Cluster performance and fairness results compared to the original paper with the variance over 5 seeds when applicable.

The results indicate that the method is able to learn fair representations and clusters. The entropy results indicate equal fair partitions, the TSNE visualizations show good clusters and the discriminator is not able to distinguish between subgroups in the cluster assignments. However, during evaluation, questions about the added fairness constraints and the experiment setup of this method arise. The optimal clustering solutions for the datasets exactly match optimal fairness. Since the datasets are fairly balanced the equal partition definition of fairness is in line with the optimal solution. Furthermore, fair clustering would perform equally good on all subgroups, which also in line with the optimal solution. Further research needs to be done to investigate this hypothesis.

6.1 What was easy and/or difficult

Based on the implementation made by the authors it was straightforward to evaluate the method on MNIST-USPS, using the provided pretrained encoders, DECs and initial centroids. These results matched the results achieved in the original paper. However, essential code and specifications were missing to extend these results to other datasets and methods. Examples include the hyperparameters for pretraining all the models, hyperparameters for running on other datasets, missing specifications for the Decoder of the VAE and the finetuning of ResNet50. This made it hard to implement the method to work for other datasets.

Despite the missing parts, the codMoste was very helpful for understanding the method in detail. Most importantly, it cleared up the workings of the structural preservation loss which was not immediately clear from the paper.

The number of different pretrained models needed was another factor that caused problems. First of all, the DFC for MNIST-USPS depends on 8 other models: 3 encoders, 2 DECs and 3 KMeans cluster initializations. This in combination with the fact that the DFC is sensitive to the quality of the pretrained models, made it hard to debug performance issues that could be caused by the model itself or any of the previous methods. And practically, it requires excellent experiment management to keep track of which models are trained with which pretrained models and parameters.

6.2 Communication with original authors

We had one-off email contact with the authors regarding questions about implementation. Specifically, we consulted them about hyperparameters for pretraining the encoders and which layers of ResNet50 to finetune. We received a helpful response in which our questions were adequately answered.

References

[1] BIEWALD, L. Experiment tracking with weights and biases, 2020. Software available from wandb.ai.

[2] CHIERICHETTI, F., KUMAR, R., LATTANZI, S., AND VASSILVITSKII, S. Fair clustering through fairlets, 2018.

[3] HE, K., ZHANG, X., REN, S., AND SUN, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 770–778.

[4] JULIA ANGWIN, JEFF LARSON, S. M., AND KIRCHNER, L. Machine bias: There is software used across the country to predict future criminals. and it's biased against blacks., 2016.

[5] KLEINBERG, J., MULLAINATHAN, S., AND RAGHAVAN, M. Inherent trade-offs in the fair determination of risk scores, 2016.

[6] LE CUN, Y., MATAN, O., BOSER, B., DENKER, J. S., HENDERSON, D., HOWARD, R. E., HUBBARD, W., JACKET, L., AND BAIRD, H. S. Handwritten zip code recognition with multilayer networks. In *[1990] Proceedings. 10th International Conference on Pattern Recognition* (1990), vol. 2, IEEE, pp. 35–40.

[7] LECUN, Y. The mnist database of handwritten digits. *http://yann.lecun.com/exdb/mnist/* (1998).

[8] LI, P., ZHAO, H., AND LIU, H. Deep fair clustering for visual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2020).

[9] SAENKO, K., KULIS, B., FRITZ, M., AND DARRELL, T. Adapting visual category models to new domains. In *European conference on computer vision* (2010), Springer, pp. 213–226.

[10] VAN DER MAATEN, L., AND HINTON, G. Visualizing data using t-sne. *Journal of machine learning research 9*, 11 (2008).

[11] XIE, J., GIRSHICK, R., AND FARHADI, A. Unsupervised deep embedding for clustering analysis, 2016.

[12] YOSINSKI, J., CLUNE, J., BENGIO, Y., AND LIPSON, H. How transferable are features in deep neural networks?, 2014.

[13] ZHANG, Z., LUO, P., LOY, C. C., AND TANG, X. Facial landmark detection by deep multi-task learning. In *European conference on computer vision* (2014), Springer, pp. 94–108.

[14] ZHAO, H., AND GORDON, G. J. Inherent tradeoffs in learning fair representations, 2020.

Appendices

A Variational Autoencoder Architecture

Layer Name	Output Size	Settings
conv1	$16 \times 32 \times 32$	filter: 3×3; stride: 1; padding: 1
bn1	$16 \times 32 \times 32$	features: 16
ReLU	$16 \times 32 \times 32$	
conv2	$32 \times 16 \times 16$	filter: 3×3; stride: 2; padding: 1
bn2	$32 \times 16 \times 16$	features: 32
ReLU	$32 \times 16 \times 16$	
conv3	$32 \times 16 \times 16$	filter: 3×3; stride: 1; padding: 1
bn3	$32 \times 16 \times 16$	features: 32
ReLU	$32 \times 16 \times 16$	
conv4	$16 \times 8 \times 8$	filter: 3×3; stride: 2; padding: 1
bn4	$16 \times 8 \times 8$	features: 16
ReLU	$16 \times 8 \times 8$	
Flatten	1024	
fc1	512	
bn5 1D	512	features: 512
ReLU	512	
mu	64	
logvar	64	

Table 2: Encoder architecture. The convolutional and batchnorm layers are standard 2D versions unless otherwise indicated. The last two layers are the output layers for respectively the meand and log variance of the distribution. These both take the last ReLU layer as input. Batch size dimension is omitted.

Layer Name	Output Size	Layer
fc1	512	
ReLU	512	
fc2	1024	
bn1 1D	1024	features: 1024
ReLU	1024	
UnFlatten	$16 \times 8 \times 8$	
convT1	$32 \times 16 \times 16$	filter: 3×3; stride: 2; padding: 1; output padding: 1
bn2	$32 \times 16 \times 16$	features: 32
ReLU	$32 \times 16 \times 16$	
convT2	$32 \times 16 \times 16$	filter: 3×3; stride: 1; padding: 1; output padding: 0
bn3	$32 \times 16 \times 16$	features: 32
ReLU	$32 \times 16 \times 16$	
convT3	$16 \times 32 \times 32$	filter: 3×3; stride: 2; padding: 1; output padding: 1
bn4	$16 \times 32 \times 32$	features: 16
ReLU	$16 \times 32 \times 32$	
convT4	$1 \times 32 \times 32$	filter: 3×3; stride: 1; padding: 1; output padding: 0
bn5	$1 \times 32 \times 32$	features: 1
Sigmoid	$1 \times 32 \times 32$	

Table 3: Decoder architecture. Transposed Convolutions, indicated with *convT*, are used as a inverse of standard convolutional layers. The architecture was chosen to resemble te inverse of the encoder. The final non linear activation is a sigmoid since this matches the loss function and original image value range. Batch size dimension is omitted.

B Discriminator Architecture

Layer Name	Output Size	Layer
fc1	32	
ReLU	32	
Dropout	32	Keep prob: 0.5
fc2	32	
ReLU	32	
Dropout	32	Keep prob: 0.5
fc3	1	
Sigmoid	1	

Table 4: Discriminator architecture. The input are cluster assignment per sample of shape $B \times C$, where B is the batch size and C the number of clusters.

C Hyperparameters

Model	Parameter	Range	Optimal VAE	Optimal ResNet
VAE	batch size	$64, 128, 256$	64	-
	learning rate	1e-4, 1e-3, 1e-2, 1e-1	0.001	-
	epochs	$20, 50, 100, 500$	50	-
DEC	batch size	$64, 128, 256$	256	128
	iterations	$10000 - 60000$	50000	20000
	learning rate	1e-4, 1e-3, 1e-2, 1e-1, $1, 10$	0.0001	0.0001
DFC	batch size	$64, 128, 256$	64	128
	iterations	$20000, 35000, 50000$	20000	20000
	learning rate	1e-4, 1e-3, 1e-2, 1e-1	0.001	0.001

Table 5: Complete list of hyperparameter search space and the optimal values. The search was performed with bayesian and a grid search for the datasets that rely on the custom VAE. Because of resource limitations the hyperparameter search for the more complex datasets that rely on ResNet, the search was manually and less extensive performed.

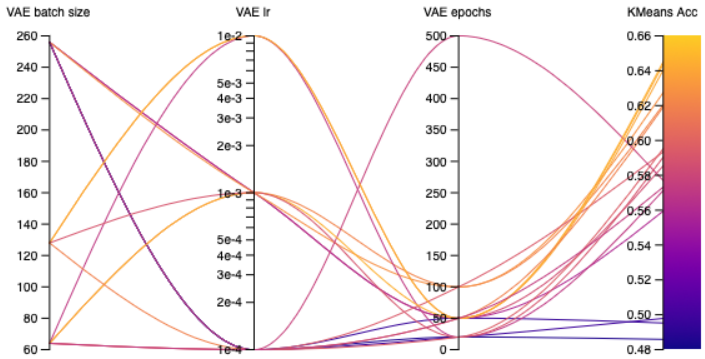

Figure 3: Overview of the results for the VAE hyperparameter search, evaluated with the clustering performance of a KMeans module on MNIST-USPS.

Model	Parameter	Value
KMeans	n init	20
	max steps	5000
	init	k-means++
	tol	1e-4
	precompute distances	auto
	algorithm	auto
TSNE	n components	2
	perplexity	30
	early exaggeration	12
	learning rate	200
	n iter	1000
	metric	euclidean
	init	random
	method	barnes hut
	angle	0.5
VAE & DEC	Adam betas	$(0.9, 0.999)$
	Adam weight decay	0
DFC	Adam betas	$(0.9, 0.999)$
	Adam weight decay	1e-4

Table 6: Complete list of default hyperparameters per model not included in the hyperparameter search.

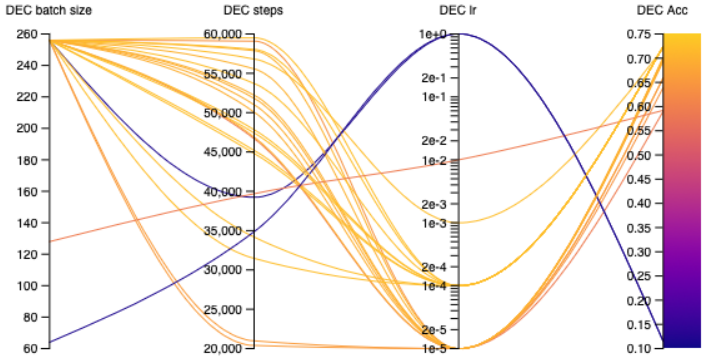

Figure 4: Overview of the results for the DEC hyperparameter search using the best performing pretrained VAE on MNIST-USPS.

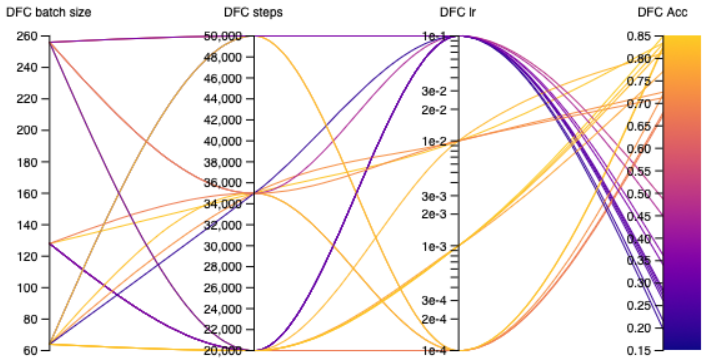

Figure 5: Overview of the results for the DFC hyperparameter search using the best performing pretrained VAE and DEC evaluated on MNIST-USPS.