Supplementary Materials

A. Choice of Hyperparameters

There are no implementations available online for ExplainGAN [39] so we had to implement it on our own. The set of hyperparameters are also not stated in the paper. We found the best set of hyperparameters though cross-validation. We weighted each loss term ($\mathcal{L}_{\text{classifier}}$, $\mathcal{L}_{\text{recon}}$, and $\mathcal{L}_{\text{prior}}$) in the objective equally with coefficient 1. Their prior loss is also comprised of multiple loss terms ($\mathcal{L}_{\text{const}}$, $\mathcal{L}_{\text{count}}$, $\mathcal{L}_{\text{smoothness}}$, and $\mathcal{L}_{\text{entropy}}$). We used the coefficient 1000 for the count loss and set the rest as 1. In addition, for the choice of κ, which controls the effect of count loss, we used 0.05 and 0.1 for MNIST and Fashion-MNIST datasets, respectively.

For CEM [7], we used the default set of hyperparameters available at the official Github repository [1] and set $\gamma = 100$, which controls the auto-encoding error. We used the implementation provided by the authors for the CVE [11] method. Their feature replacement search occurs at the last convolutional layer of the classifier. In order to generate comparable CFs with other baselines, we slightly changed the objective in both CEM and CVE methods and set the target to a user-specified class rather than *maximum-non-query* class.

We added the PGD targeted adversarial attack [25] as a baseline. We used the *torchattacks*[2] library for doing the attacks. We set the step size $\alpha = 1/255$, maximum number of steps to 1000, and maximum step sizes of $\epsilon = 64/255$ and $\epsilon = 72/255$ for MNIST and Fashion-MNIST datasets, respectively.

Each loss term in the main objective of C3LT is scaled by a coefficient which the values are obtained through cross-validation:

$$\mathcal{L}_{c3lt} = \mathcal{L}_{cls} + \alpha\mathcal{L}_{prx} + \beta\mathcal{L}_{cyc} + \gamma\mathcal{L}_{adv} \tag{11}$$

where we set $\{\alpha = 0.1, \beta = 0.1, \gamma = 0.001\}$ for both the MNIST and Fashion-MNIST datasets. In addition, we found one non-linear step to be sufficient in our experiments and set $n = 1$.

We used the PyTorch [31] framework to implement and evaluate all methods (including C3LT) and deep neural networs, except for the CEM which we used the original implementation in Tensorflow [1]. Across all methods, we used the same pretrained classifier for both MNIST (99.4% accuracy) and Fashion-MNIST (91.5%) datasets.

[1] https://github.com/IBM/Contrastive-Explanation-Method

[2] https://github.com/Harry24k/adversarial-attacks-pytorch

B. Computation Time Comparison

It is ideal that the CF examples are generated on the fly. This is particularly helpful when users and machine explanations interact. Many approaches, including CEM and CVE, generate CF explanations by solving iterative optimization problems and there are no training phase. Hence, it is not a surprise such methods are not *fast* and cannot be used for real-time applications. On the other hand, our method generate CFs orders of magnitude faster than iterative methods. This is mainly due to the fact that our method only does a forward pass in the C3LT pipeline during inference time. We present the average computation time (per sample) to generate CF explanations from our method and baselines for the MNIST dataset in Table. 4. ExpGAN is showing comparable results since their approach also does a forward pass at inference time. However, it is slightly slower as their generator has multiple heads while ours does not. We used the same batch size of 256 for evaluating our method and ExpGAN. CVE and CEM generate explanations one sample at a time. To run the experiments, we used a HP Z640 Workstation with a single NVIDIA GeForce RTX 2080-Ti GPU.

C. Ablation Study

To analyze the contribution of each loss term in $\mathcal{L}_{c3lt}$, we conduct an ablation study. We respectively add the $\mathcal{L}_{prx}$, $\mathcal{L}_{cyc}$, and $\mathcal{L}_{adv}$ loss terms to the $\mathcal{L}_{cls}$ and evaluate the generated CFs in terms of the metrics explained in the paper. Table 5 shows the obtained results from our ablation study on MNIST dataset. Minimizing the $\mathcal{L}_{cls}$ generates images that are in the CF class. However, it does not consider minimal perturbations to the input in order to change the decision of the classifier, *i.e.*, the input images and the obtained CFs are distant. This is reflected in the proximity metric ($Prox$). Adding the $\mathcal{L}_{prx}$ encourages such minimal changes and improves the proximity score while maintaining almost perfect validity. As one can expect, this improves the COUT metric as well. Adding the $\mathcal{L}_{cyc}$ further regularizes the training and helps with learning more accurate transformations. This further improves the proximity and COUT metrics. Finally, adding the $\mathcal{L}_{adv}$ helps with improving the realism metrics (IM1, IM2, FID, and KID). This ensures the generated CFs stay close to the data manifold, resulting in changes that are actionable and sensible to humans.

D. Distinction from CycleGAN

Here, we first evaluated our method against CycleGAN [50]. Then, we elaborate on the similarities of C3LT and CycleGAN and how they distinct from each other. Finally, we showcase debugging a classifier using C3LT where methods such as CycleGAN are not useful.

Table 4. **Computation Time Comparison of CFs**. This table shows the average computation time (per sample) to generate CF examples in seconds for the MNIST dataset. C3LT is relatively faster than ExpGAN while being significantly faster than CVE and CEM — as they solve iterative optimization problems to generate CFs. At inference time, our method only does a forward pass through the C3LT pipeline to generate CFs. This makes our method suitable for CF explanation generation on the fly.

Methods	ExpGAN [39]	CEM [7]	CVE [11]	C3LT (ours)
Time (*sec*)↓	$1.28\text{e}{-5}$	68.34	$2.91\text{e}{-2}$	$\mathbf{9.23e{-6}}$

Table 5. **Ablation Study**. In this table, we provide the quantitative results obtained from the ablation study of the C3LT objective. We respectively add the $\mathcal{L}_{prx}$, $\mathcal{L}_{cyc}$, and $\mathcal{L}_{adv}$ loss terms to the $\mathcal{L}_{cls}$ and evaluate the generated CFs using the CF evaluation metrics.

Lost Terms	$COUT\uparrow$	$IM1\downarrow$	$IM2\times 10\downarrow$	$FID\downarrow$	$KID\times 1\text{e}3\downarrow$	$Prox\downarrow$	$Val\uparrow$
$\mathcal{L}_{cls}$	0.897	0.47	0.29	40.84	29.22	0.113	**1.0**
$\mathcal{L}_{cls}+\mathcal{L}_{prox}$	0.935	0.65	0.34	33.40	22.36	0.076	0.998
$\mathcal{L}_{cls}+\mathcal{L}_{prox}+\mathcal{L}_{cyc}$	0.943	0.78	0.38	29.95	18.65	**0.069**	0.998
$\mathcal{L}_{cls}+\mathcal{L}_{prox}+\mathcal{L}_{cyc}+\mathcal{L}_{adv}$ (C3LT)	**0.948**	**0.70**	**0.36**	**22.83**	**13.39**	0.072	0.999

CycleGAN learns image-to-image translation using generative adversarial training. We used the images from the query and CF classes as the input and output image domains. We used the official implementation of CycleGAN[3] and trained translation functions on MNIST and Fashion-MNIST datasets. Fig. 6 visually compares the CF examples obtained from CycleGAN and C3LT on both datasets and various pairs. The generated CFs from C3LT are more realistic and sharp while CycleGAN results are often blurry and scattered with meaningless perturbations across the image (e.g. 3 to 8). In addition, the C3LT translations are more proximal to the original input image (e.g. sneaker to boot).

Table 6 shows quantitative comparison of CycleGAN and C3LT in terms of CF metrics on MNIST dataset. Following the insights obtained from the visual comparison of the CFs in Fig. 6, Table 6 corroborates that the CF examples from C3LT are more realistic and have higher quality. In addition, the COUT score obtained in this comparison shows that the C3LT generates more valid and sparse explanations.

CycleGAN [50] learns unpaired image translation using a cycle-consistent generative adversarial training. While the cycle-consistency in the C3LT is inspired by it, there are two main differences that separates our work. First, that our cycle-consistency is in the latent space of a given (pre-trained) generator and the transformations are occurring in the latent space, rather than direct image-to-image translation. This is favorable as our method can be easily plugged into state-of-the-art pretrained generative models (GANs, VAEs, etc.) and discard training them from scratch. Second, the CycleGAN is *not* explaining a classifier. Indeed, CycleGAN uses two different discriminators to keep the translated images close to the data manifold of each target class. However, the main goal of this paper is to explain a given classifier through CF explanations. In the follwoing, we show that C3LT can be used for debugging a *faulty* classifier while methods such as CycleGAN are not helpful.

[3]https://github.com/junyanz/CycleGAN

We use C3LT to provide explanations for a *faulty* classifier. Here, we simply rig a classifier by depriving it from seeing examples from a specific class during the training. Put it differently, we train a classifier that lacks knowledge regarding a specific class while it is having a reasonably well performance on the rest the classes. To that end, we trained a classifier (identical to the one used for experiments) on MNIST dataset while discarding the training examples from class 9. This classifier obtains $\sim 89\%$ test accuracy — only missing test samples from the *left-out* class 9. We then attempt to explain the decision of this classifier using C3LT as shown in Fig. 7. Choosing the class 4 for the query images, we set classes 9 (left-out class) and 1 (non-left-out class) as the target for the CF explanations. As one might expect, the CFs for the left-out class are not interpretable and meaningful which emphasizes the classifier lacks knowledge regarding the target class. On the other hand, when choosing the non-left-out class 1, the CF explanations are intuitive and might be helpful to a user, whereas GAN-type approaches such as CycleGAN will just continue to generate normal digits without using the classifier. This is a simple scenario for understanding the weakness of a classifier; however, it emphasises the substantial differences between C3LT and methods such as CycleGAN. While our method can explain *any* classifier, CycleGAN and other GAN-type methods are not of use.

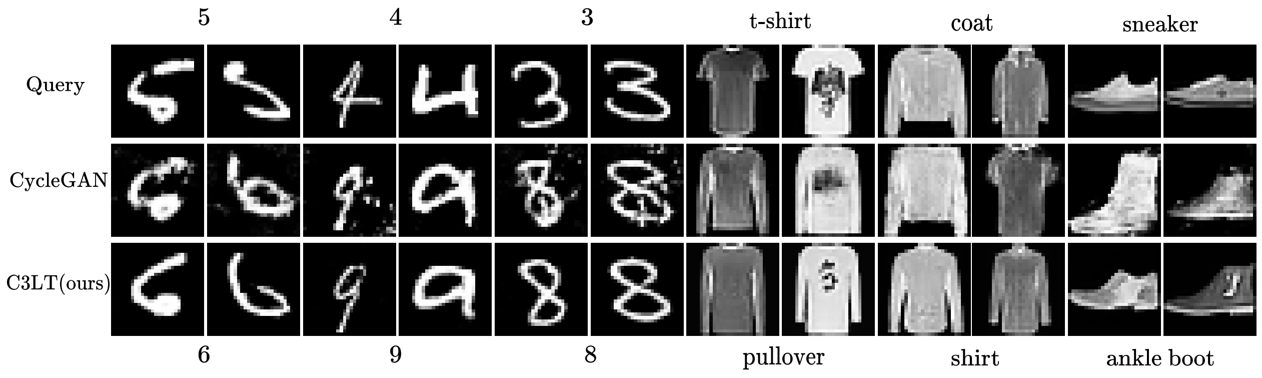

Figure 6. **Visual comparison of C3LT and CycleGAN**. This figure compares the CF examples generated by C3LT and CycleGAN on MNIST and Fashion-MNIST datasets. CF examples obtained from CycleGAN are often blurry with scattered perturbation with respect to the query image while the CF images from C3LT are more realistic, sharp, and close to the original query image.

Table 6. **Quantitative Comparison of C3LT against CycleGAN**. We compare the CF obtained from CycleGAN and our method in terms of CF metrics. CF examples generated from C3LT are more realistic and have higher quality.In addition, the COUT score obtained in this experiment shows that the C3LT generates more valid and sparse explanations than CycleGAN.

Metric / Method	$COUT \uparrow$	$IM1 \downarrow$	$IM2 \times 10 \downarrow$	$FID \downarrow$	$KID \times 1e3 \downarrow$	$Prox \downarrow$	$Val \uparrow$
C3LT(ours)	**0.948**	**0.70**	**0.36**	**22.83**	**13.39**	**0.072**	**0.999**
Cycle-GAN [50]	0.894	0.716	0.49	43.4	41.61	0.089	0.988

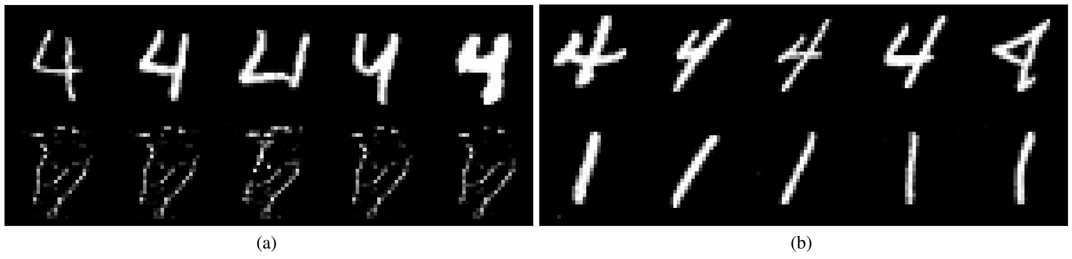

Figure 7. **Debugging a faulty Classifier**. Here, we showcase the capability of C3LT in debugging a *rigged* classifier while methods such as CycleGAN are not helpful. We choose images from class 4 as the query images (top row) and set the a) *left-out* b)*non-left-out* class as the target for the CF examples (bottom row). While the CFs for the left-out class are not interpretable, the CFs for the non-left-out class are intuitive. GAN-type approaches such as CycleGAN do not interpret different classifiers and would generate regular digits 9 and 1 in either case, respectively.

E. Traversal in the Latent Space

”Does C3LT lead to disentangled transformations in the latent space?” Normal VAE/GANs do not generate disentangled latents. Fig. 8 is a violin plot showing the mean absolute difference in latent dimensions between the original and CF images, for three class pairs from MNIST. It shows most latent dimensions are changed (with average magnitude of 0.2). Note our main goal is to build a method to generate realistic and high-resolution CF images for explaining classifiers, so sparsity of latent traversal is interesting future work but orthogonal to this goal.

However, when steering from the input to the CF in the latent space, we observe meaningful traversal. Fig. 9 illustrates this for (pullover, t-shirt) and (4, 9) class pairs with $n = 3$ discrete steps (see Eq.1 in the paper).

F. Additional Visual Examples

In the following, we show more CFs generated from our method and baselines for both MNIST and Fashion-MNIST datasets.

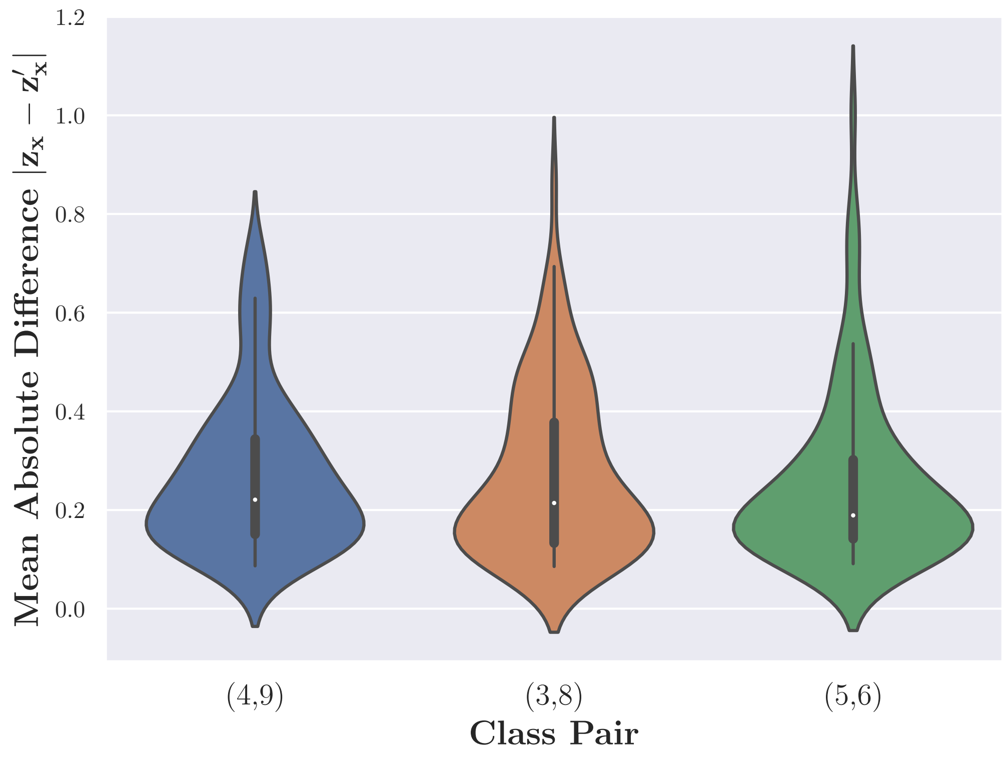

Figure 8. Mean absolute difference of the input and CF latent codes using C3LT for class pairs from MNIST. It can be observed the that the learned transformations are not sparse.

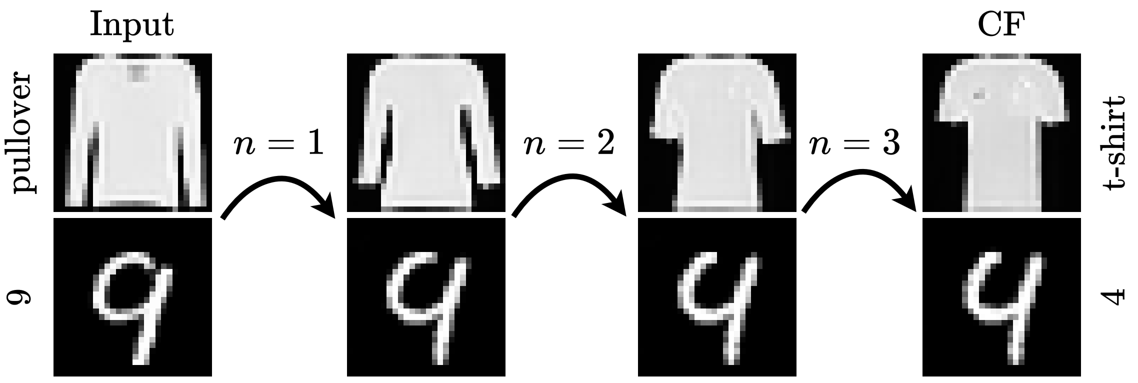

Figure 9. Traversal with $n = 3$ steps in the latent space of the generator going from the input to the CF example.

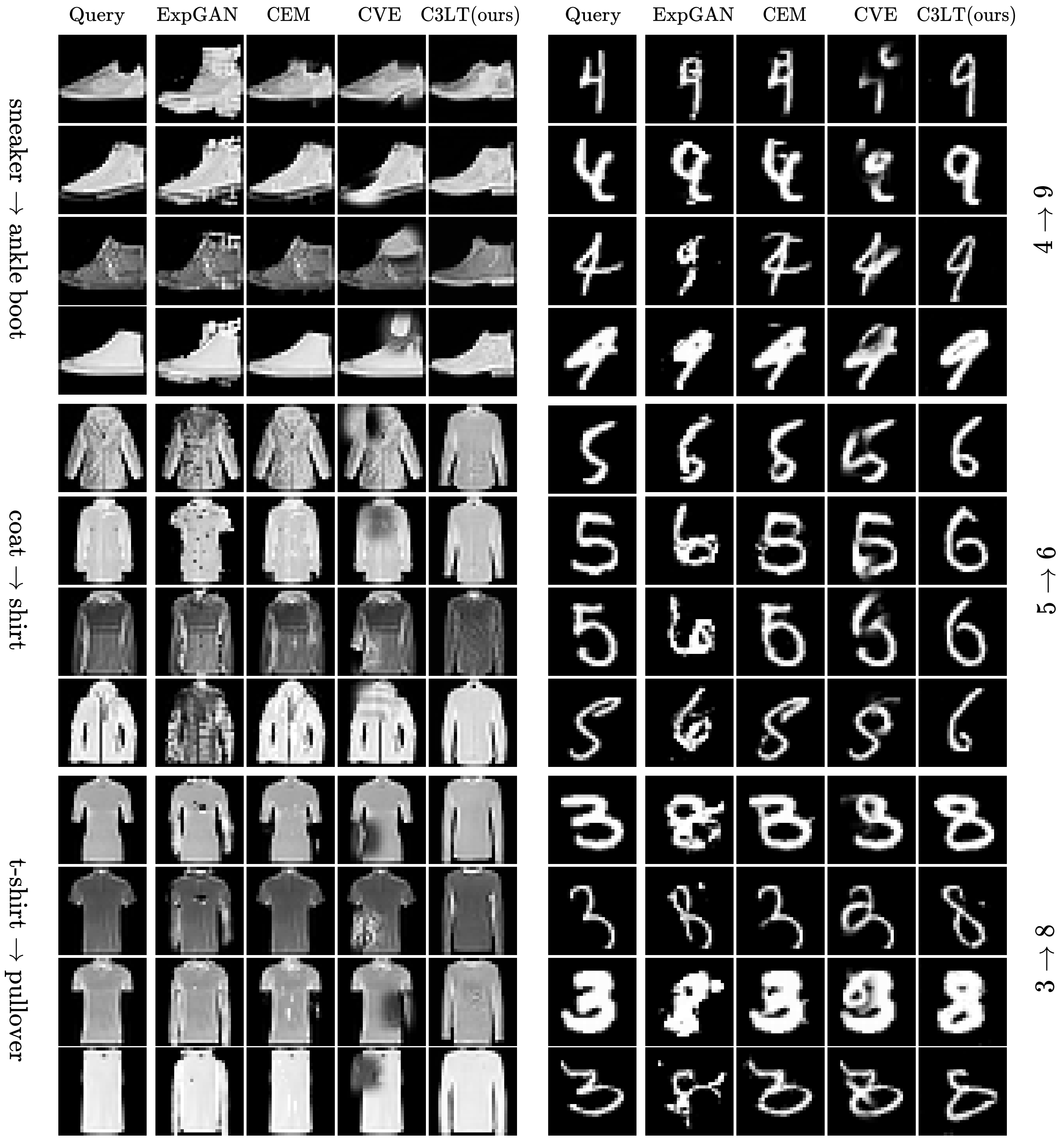

Figure 10. **Visual Comparison of the CFs**. This figure illustrated the generated CFs from C3LT, ExplainGAN, CEM, and CVE for both MNIST and Fashion-MNIST datasets. For MNIST, we show the (3,8), (4,9), and (5,6) pairs. For Fashion-MNIST, the pairs are (t-shirt, pullover), (coat, shirt), and (sneaker, ankle boot). It can be noted thatthe generated CFs from the CEM and CVE to be adversarial and off the data manifold. Compared to the ExpGAN, the generated CFs from our method are consistently more realistic and interpretable.