Appendix

This appendix is organized as follows:

- Section A.1 proves the theorem in Section 3.4 as a corollary of [3];
- Section A.2 gives the implementation details of our model in ZSL (Section A.2.1) and OSR (Section A.2.2);
- Section A.3 includes additional experimental results; Specifically, Section A.3.1 shows additional results on two-stage inference and a comparison between entangled and disentangled model in ZSL Accuracy. Section A.3.2 shows additional results for OSR, with the details of the 5 splits that we used and the F1 score for each split, closed-set accuracy, a comparison between entangled and disentangled model and more qualitative results.

A.1. Proof of the Theorem in 3.4

Let $V = (Z, Y)$ and V takes values from the space $\mathcal{V}$. For class attribute Y that is a K-dimensional real vector, denote $\mathcal{E} = \{e_1, \ldots, e_K\}$ as the subset of dimensions spanned by the class attribute Y, and $\bar{\mathcal{E}}$ as those by the sample attribute Z. Therefore, $\mathcal{V}^{\mathcal{E}}$ and $\mathcal{V}^{\bar{\mathcal{E}}}$ represent the space of class attribute Y and sample attribute Z, respectively. We use $g : \mathcal{V} \to \mathcal{X}$ to denote the endogenous mapping to the feature space $\mathcal{X}$. Note that this g corresponds to sampling from $P_\theta(X|Z, Y)$ in our GCM. We will introduce the concept of embedded GCM to facilitate the proof.

Definition (Embedded GCM). *We say that a GCM is embedded if $g : \mathcal{V} \to \mathcal{X}$ is a continuous injective function with continuous inversion g^{-1}.*

Using the results in [1], if V is compact (*i.e.*, bounded), the GCM is embedded if and only if g is injective. Though we implement our GCM using VAE, whose latent space is not compact, it is shown that restricting a VAE to a product of compact intervals that covers most of the probability mass (using KL-divergence in the objective) will result in an embedded GCM that approximates the original one for most samples [3]. Moreover, $P_\theta(X|Z, Y)$ are implemented using deterministic mappings in our model (see Section A.2.1, A.2.2), which are indeed injective as shown in [12]. Therefore, without loss of generality, our GCM can be considered as embedded. We will give the formal definition of intrinsic disentanglement, which can be used to show that group disentanglement of Z and Y leads to faithfulness.

Definition (Intrinsic Disentanglement). *In a GCM, the endomorphism $T : \mathcal{X} \to \mathcal{X}$ is intrinsically disentangled with respect to the subset $\mathcal{E}$ of endogenous variables, if there exists a transformation T' affecting only variables indexed by $\mathcal{E}$, such that for any $\mathbf{v} \in \mathcal{V}$,*

$$T(g(\mathbf{v})) = g(T'(\mathbf{v})). \tag{1}$$

Now we will first establish the equivalence between intrinsic disentanglement and faithfulness using the theorem below.

Theorem (Intrinsic Disentanglement and Faithfulness). *The counterfactual mapping $X_{\mathbf{y}}[z(X)]$ is faithful if and only if it is intrinsically disentangled with respect to the subset $\mathcal{E}$.*

To prove the above theorem, one conditional is trivial: if a transformation is intrinsically disentangled, it is by definition an endomorphism of $\mathcal{X}$ so the counterfactual mapping must be faithful. For the second conditional, let us assume a faithful counterfactual mapping $X_{\mathbf{y}}[z(X)]$. Based on the three steps of computing counterfactuals and the embedding property discussed earlier, the counterfactual mapping can be decomposed as:

$$X_{\mathbf{y}}[z(X)] = g \circ T' \circ g^{-1}(X), \tag{2}$$

where $\circ$ denotes function composition, T' is the counterfactual transformation of V as defined in Section 3.2, where Z is kept as $Z = z(X)$ and Y is set as $\mathbf{y}$. Now for any $\mathbf{v} \in \mathcal{V}$, the quantity $X_{\mathbf{y}}[z(g(\mathbf{v}))]$ can be similarly decomposed as:

$$X_{\mathbf{y}}[z(g(\mathbf{v}))] = g \circ T' \circ g^{-1} \circ g(\mathbf{v}) = g \circ T'(\mathbf{v}). \tag{3}$$

Since T' is a transformation that only affects variables in $\mathcal{E}$ (*i.e.*, Y), we show that faithful counterfactual transformation $X_{\mathbf{y}}[z(X)]$ is intrinsically disentangled with respect to $\mathcal{E}$.

In our work, we define group disentanglement of Z and Y as intrinsic disentanglement with respect to the set of variables in Y. We used the sufficient condition by learning a GCM where Y and Z are group disentangled, such that when we fix Z and only change Y, the resulting generation lies in $\mathcal{X}$ according to the theorem that we just proved. We also exploited the necessary condition by training for faithfulness through the WGAN loss, which in turn encourages the group disentanglement of Z and Y.

A.2. Implementation Details

A.2.1. ZSL

Our GCM implementation is based on the generative models in TF-VAEGAN [10]. Besides $P_\theta(X|Z, Y)$ and $Q_\phi(Z|X)$ that is common in a VAE-GAN [7], it additionally implements $Q_\psi(Y|X)$ and a feedback module, which takes the intermediate layer output from the network in $Q_\psi(Y|X)$ as input and generate a feedback to assist the generation process $P_\theta(X|Z, Y)$. The rest of the section will detail the network architecture for each component, followed by additional training and inference details supplementary to Section 3.4 and 3.3, respectively.

Sample Attribute Z. The dimension of sample attribute Z is set to be the same as that of the class attribute for each dataset. For example, in CUB [16], the dimension of Z is set as 312. $P(Z)$ is defined as $\mathcal{N}(\mathbf{0}, \mathbf{I})$ for all datasets.

Decoder $P_\theta(X|Z,Y)$. The module that implements this conditional distribution is commonly known as the decoder in literature [13, 10]. We implemented $P_\theta(X|Z,Y)$ with Deep Gaussian Family $\mathcal{N}(\mu_D(Z,Y), \mathbf{I})$ with its variance fixed and mean generated from a network μ_D. μ_D was implemented with a Multi-Layer Perceptron (MLP) with two layers and LeakyReLU activation (alpha=0.2) in between. The input to the MLP is the concatenated Z and Y. The hidden layer size is set as 4,096. The MLP outputs a real vector of size 2,048 (same as that of X) and the output goes through a Sigmoid activation to produce the mean of $P_\theta(X|Z,Y)$.

Encoder $Q_\phi(Z|X)$. For convenience, we refer to $Q_\phi(Z|X)$ as the encoder. We implemented $Q_\phi(Z|X)$ with $\mathcal{N}(\mu_E(X), \sigma_E^2(X))$, where $\mu_E(X), \sigma_E^2(X)$ are neural networks that share identical architecture. Specifically, they are 3-layer MLP with LeakyReLU activation (alpha=0.2) that takes X as input and outputs a vector with the same dimension as Z. The first hidden layer size is set as 4,096 and the second hidden layer size is set as two times of the dimension of Z. Note that in the original TF-VAEGAN implementation, the encoder additionally conditions on Y. We argue that this may cause the encoded Z to contain information about Y, undermining the disentanglement. Hence we make the encoder conditioned only on X.

Regressor $Q_\psi(Y|X)$. It is a 2-layer MLP with LeakyReLU activation (alpha=0.2). The hidden layer size is set as 4,096.

Discriminator $D(X,Y)$. It is a 2-layer MLP with LeakyReLU activation (alpha=0.2). It takes the concatenated X and Y as input and outputs a single real value. The hidden layer size is set as 4,096.

Feedback Module. It is a 2-layer MLP with LeakyReLU activation (alpha=0.2). The hidden layer output of the regressor is sent to the feedback module as input. This module generates a real vector with 4,096 dimensions, which is added to the hidden layer output of μ_D as feedback signal.

Datasets. The details of ZSL datasets are given in Table A1.

Dataset	Granularity	Total	$\lvert\mathcal{S}\rvert$	$\lvert\mathcal{U}\rvert$
CUB [16]	Fine	11,788	150	50
SUN [18]	Fine	14,340	645	72
AWA2 [17]	Coarse	37,322	40	10
aPY [4]	Coarse	12,051	20	12

Table A1: Information on ZSL datasets.

Training. The networks are trained in an iterative fashion. First, all networks except the decoder are optimized. Then the discriminator is frozen and all other networks are optimized. We followed the optimization settings in TF-VAEGAN. Specifically, the Adam [5] optimizer is used with learning rate set as $1e^{-4}$ in CUB [16], $1e^{-5}$ in AWA2 [17], $1e^{-3}$ in SUN [18] and $1e^{-5}$ in aPY [4]. For the hyperparameters in our counterfactual-faithful training, we used $\beta = 6.0, \nu = 1.0$ for CUB, $\beta = 6.0, \nu = 1.0$ for AWA2, $\beta = 4.0, \nu = 1.0$ for SUN and $\beta = 6.0, \nu = 0$ for aPY. On CUB, AWA2, and SUN, we additionally used annealing on the KL divergence loss, where β is initially set as 0 and linearly increased to the set value over 40 epochs.

Inference. In ZSL, we train a linear classifier with one fully-connected layer using the Adam optimizer. On CUB, the classifier was trained for 15 epochs with learning rate as $1e^{-3}$. On AWA2, it was trained for 3 epochs with learning rate as $1e^{-3}$. On SUN, it was trained for 6 epochs with learning rate as $5e^{-4}$. On aPY, it was trained for 3 epochs with learning rate as $1e^{-3}$. After training, the classifier was used for inference following the decision rule introduced in Section 3.3.

A.2.2. OSR

Our proposed GCM-CF is implemented based on the architecture of CGDL [14]. Notice that the original CGDL doesn't distinguish sample attribute Z and class attribute Y explicitly. To keep consistent with the ZSL model and follow the common VAE-GAN architecture, here we revise the encoder to model Z and Y respectively.

Encoder $Q_\phi(Z|X)$. Given an actual image $X = \boldsymbol{x}$, we follow [14] to implement $Q_\phi(Z|X)$ with the probabilistic ladder architecture to extract the high-level abstract latent features $\boldsymbol{z}$. In detail, the l-th layer in the ladder encoder is expressed as:

$$\begin{aligned}
\boldsymbol{x}_l &= \text{Conv}(\boldsymbol{x}_{l-1}) \\
\boldsymbol{h}_l &= \text{Flatten}(\boldsymbol{x}_l) \\
\boldsymbol{\mu}_l &= \text{Linear}(\boldsymbol{h}_l) \\
\boldsymbol{\sigma}_l^2 &= \text{Softplus}(\text{Linear}(\boldsymbol{h}_l))
\end{aligned}$$

where `Conv` is a convolutional layer followed by a batch-norm layer and a PReLU layer, `Flatten` is a linear layer to flatten 2-dimensional data into 1-dimension, `Linear` is a single linear layer and `Softplus` applies $\log(1+\exp(\cdot))$ non-linearity to each component of its argument vector. The latent representation $Z = \boldsymbol{z}$ can be obtained as:

$$\begin{aligned}
\boldsymbol{\mu_z}, \boldsymbol{\sigma_z^2} &= \text{LadderEncoder}(\boldsymbol{x}), \\
\boldsymbol{z} &= \mu_z + \sigma_z \odot \mathcal{N}(0, \mathbf{I}),
\end{aligned} \tag{4}$$

where $\odot$ is the element-wise product.

Decoder $P_\theta(X|Z,Y)$. Given the latent sample attribute $Z = \boldsymbol{z}$ and the class attribute $Y = \boldsymbol{y}$, the l-th layer of

the ladder decoder is expressed as follows:

$$\begin{aligned}
\tilde{\boldsymbol{c}}_{l+1} &= \text{Unflatten}(\boldsymbol{t}_{l+1}) \\
\tilde{\boldsymbol{x}}_{l+1} &= \text{ConvT}(\tilde{\boldsymbol{c}}_{l+1}) \\
\tilde{\boldsymbol{h}}_{l+1} &= \text{Flatten}(\tilde{\boldsymbol{x}}_{l+1}) \\
\tilde{\boldsymbol{\mu}}_{l} &= \text{Linear}(\tilde{\boldsymbol{h}}_{l+1}) \\
\tilde{\boldsymbol{\sigma}}_{l}^2 &= \text{Softplus}(\text{Linear}(\tilde{\boldsymbol{h}}_{l+1})) \\
\boldsymbol{t}_{l} &= \tilde{\boldsymbol{\mu}}_{l} + \tilde{\boldsymbol{\sigma}}_{l}^2 \odot \boldsymbol{\epsilon}
\end{aligned}$$

where `ConvT` is a transposed convolutional layer and `Unflatten` is a linear layer to convert 1-dimensional data into 2-dimension. Note that the input $\boldsymbol{t}$ of the top decoder layer is the concatenation of $\boldsymbol{z}$ and $\boldsymbol{y}$. Overall, the reconstructed image $\tilde{\boldsymbol{x}}$ can be represented as:

$$\tilde{\boldsymbol{x}} = \text{LadderDecoder}(\boldsymbol{z}, \boldsymbol{y}). \tag{5}$$

For more details please refer to [14].

Known Classifier. The known classifier is a Softmax Layer taking the one-hot embedding $\boldsymbol{y}$ as the input and produces the probability distribution over the known classes.

Datasets. The details of OSR datasets are given in Table A2.

Dataset	Image Size	Classes	Train	Test
MNIST [8]	28×28	10	60,000	10,000
SVHN [11]	32×32	10	73,257	26,032
CIFAR10 [6]	32×32	10	50,000	10,000
CIFAR100 [6]	32×32	100	50,000	10,000

Table A2: Information on OSR datasets.

Training. The network is trained in an end-to-end fashion. We directly follow the optimization settings and hyperparameters of ladder architecture in CGDL [14]. For the hyperparameters in our counterfactual-faithful training, we used $\beta = 1.0, \nu = 10.0$ for MNIST, $\beta = 1.0, \nu = 2.0$ for SVHN, $\beta = 6.0, \nu = 1.0$ for CIFAR10, $\beta = 1.0, \nu = 20.0$ for CIFARAdd10, $\beta = 1.0, \nu = 1.0$ for CIFARAdd50. Note that we didn't use the loss L_F in OSR task.

Inference. When training is completed, we follow [14] to use the reconstruction errors and a multivariate Gaussian model to judge the unseen samples. The threshold τ_l is set to 0.9 for MNIST, 0.6 for SVHN, 0.9 for CIFAR10, 0.8 for CIFARAdd10 and 0.5 for CIFARAdd50. More details about the multivariate Gaussian model please refer to [14].

A.3. Additional Results

A.3.1. ZSL

Stage-One Binary Classifier. We extend the results in Table 5 by showing comparison of the two-stage inference performance on CUB [16], SUN [18] and aPY [4] dataset. Our GCM-CF improves all of them and outperforms the current SOTA ZSL method TF-VAEGAN as a binary unseen/seen classifier.

Dataset	CUB [16]					
Stage 1 / Stage 2	**TF-VAEGAN [10]**			**GCM-CF (Ours)**		
	U	S	H	U	S	H
RelationNet [15]	40.5	65.3	50.0	47.7	57.6	**52.2**
CADA-VAE [13]	43.2	63.4	51.4	51.4	57.6	**54.3**
LisGAN [9]	41.1	66.0	50.7	47.9	58.1	**52.5**
TF-VAEGAN [10]	50.8	64.0	56.6	55.4	60.0	**57.6**
Dataset	SUN [18]					
Stage 1 / Stage 2	**TF-VAEGAN [10]**			**GCM-CF (Ours)**		
	U	S	H	U	S	H
RelationNet [15]	30.8	23.0	26.3	37.2	21.9	**27.6**
CADA-VAE [13]	37.6	39.3	38.4	44.6	37.6	**40.8**
LisGAN [9]	36.3	41.7	38.8	43.0	38.9	**40.8**
TF-VAEGAN [10]	41.7	39.9	40.8	47.9	37.8	**42.2**
Dataset	aPY [4]					
Stage 1 / Stage 2	**TF-VAEGAN [10]**			**GCM-CF (Ours)**		
	U	S	H	U	S	H
RelationNet [15]	31.5	63.3	42.1	34.6	56.6	**43.0**
CADA-VAE [13]	31.1	64.8	41.9	35.0	57.2	**43.5**
LisGAN [9]	31.2	64.6	42.0	34.7	57.3	**43.2**
TF-VAEGAN [10]	33.1	64.2	43.7	37.1	56.8	**44.9**

Table A3: Supplementary to Table 5. Comparison of the two-stage inference performance on CUB [16], SUN [18] and aPY [4] using TF-VAEGAN [10] and our GCM-CF as the stage-one binary classifier.

Dataset	**Entangled**			**Disentangled**		
	U	S	H	U	S	H
CUB [16]	71.6	15.8	25.9	61.0	59.7	**60.3**
AWA2 [17]	70.5	27.9	40.0	60.4	75.1	**67.0**
SUN [18]	62.9	17.5	27.4	47.9	37.8	**42.2**
aPY [4]	42.4	14.4	21.5	37.1	56.8	**44.9**

Table A4: Comparison of ZSL Accuracy using an entangled model without using the proposed counterfactual-faithful training and the disentangled model with the proposed training.

Effect of Disentanglement. To show that the quality of disentangling Z and Y is the key bottleneck, we compared an entangled model (without counterfactual-faithful training) and the disentangled model on ZSL Accuracy and the results are shown in Table A4. Notice that the entangled model has a much lower S. This is because the training is conducted on the seen-classes and the encoded Z is entangled with seen-classes attributes. Therefore the generated counterfactuals are biased towards the seen-classes, *i.e.*, the green counterfactual samples in Figure 2c are closer to true samples from seen-classes, pushing the classifier boundary towards seen-classes and increasing the recall of the unseen-class by sacrificing that of the seen.

A.3.2. OSR

5 Splits Results. In our main paper we have argued that the common evaluation setting of averaging F1 scores over 5 random splits can result in a large variance in the F1 score. Here we additionally report the split details in Table A5 and

the results on all splits in Tabel A6. Note that since the official code of CGDL [14] is not complete, we implemented the code of dataloader, computing F1 score and set part of parameters. The 5 seeds (*i.e.*, 5 splits) are randomly chosen without any selection.

Split	1 (seed: 777)	
Dataset	Seen	Unseen
MNIST	3,7,8,2,4,6	0,1,5,9
SVHN	3,7,8,2,4,6	0,1,5,9
CIFAR10	3,7,8,2,4,6	0,1,5,9
CIFARAdd10	0,1,8,9	27, 46, 98, 38, 72, 31, 36, 66, 3, 97
CIFARAdd50	0,1,8,9	27, 46, 98, 38, 72, 31, 36, 66, 3, 97, 75, 67, 42, 32, 14, 93, 6, 88, 11, 1, 44, 35, 73, 19, 18, 78, 15, 4, 50, 65, 64, 55, 30, 80, 26, 2, 7, 34, 79, 43, 74, 29, 45, 91, 37, 99, 95, 63, 24, 21
Split	2 (seed: 1234)	
Dataset	Seen	Unseen
MNIST	7,1,0,9,4,6	2,3,5,8
SVHN	7,1,0,9,4,6	2,3,5,8
CIFAR10	7,1,0,9,4,6	2,3,5,8
CIFARAdd10	0,1,8,9	98, 46, 14, 1, 7, 73, 3, 79, 93, 11
CIFARAdd50	0,1,8,9	98, 46, 14, 1, 7, 73, 3, 79, 93, 11, 37, 29, 2, 74, 91, 77, 55, 50, 18, 80, 63, 67, 4, 45, 95, 30, 75, 97, 88, 36, 31, 27, 65, 32, 43, 72, 6, 26, 15, 42, 19, 34, 38, 66, 35, 21, 24, 99, 78, 44
Split	3 (seed: 2731)	
Dataset	Seen	Unseen
MNIST	8,1,6,7,2,4	0,3,5,9
SVHN	8,1,6,7,2,4	0,3,5,9
CIFAR10	8,1,6,7,2,4	0,3,5,9
CIFARAdd10	0,1,8,9	79, 98, 67, 7, 77, 42, 36, 65, 26, 64
CIFARAdd50	0,1,8,9	79, 98, 67, 7, 77, 42, 36, 65, 26, 64, 66, 73, 75, 3, 32, 14, 35, 6, 24, 21, 55, 34, 30, 43, 93, 38, 19, 99, 72, 97, 78, 18, 31, 63, 29, 74, 91, 4, 27, 46, 2, 88, 45, 15, 11, 1, 95, 50, 80, 44
Split	4 (seed: 3925)	
Dataset	Seen	Unseen
MNIST	7,3,8,4,6,1	0,2,5,9
SVHN	7,3,8,4,6,1	0,2,5,9
CIFAR10	7,3,8,4,6,1	0,2,5,9
CIFARAdd10	0,1,8,9	46, 77, 29, 24, 65, 66, 79, 21, 1, 95
CIFARAdd50	0,1,8,9	46, 77, 29, 24, 65, 66, 79, 21, 1, 95, 36, 88, 27, 99, 67, 19, 75, 42, 2, 73, 32, 98, 72, 97, 78, 11, 14, 74, 50, 37, 26, 64, 44, 30, 31, 18, 38, 4, 35, 80, 45, 63, 93, 34, 3, 43, 6, 55, 91, 15
Split	5 (seed: 5432)	
Dataset	Seen	Unseen
MNIST	2,8,7,3,5,1	0,4,6,9
SVHN	2,8,7,3,5,1	0,4,6,9
CIFAR10	2,8,7,3,5,1	0,4,6,9
CIFARAdd10	0,1,8,9	21, 95, 64, 55, 50, 24, 93, 75, 27, 36
CIFARAdd50	0,1,8,9	21, 95, 64, 55, 50, 24, 93, 75, 27, 36, 73, 63, 19, 98, 46, 1, 15, 72, 42, 78, 77, 29, 74, 30, 14, 38, 80, 45, 4, 26, 31, 11, 97, 7, 66, 65, 99, 34, 6, 18, 44, 3, 35, 88, 43, 91, 32, 67, 37, 79

Table A5: The detailed label splits of 5 random seeds

Closed Set Results. Closed-Set Accuracy is the standard supervised classification accuracy on the seen-classes with open set detection disabled. As shown in Table A7, the network were trained without any large degradation in closed set accuracy from the plain CNN.

Split	1				
Method	MNIST	SVHN	CIFAR10	C+10	C+50
Softmax	76.26	73.06	69.81	77.87	65.78
OpenMax [2]	83.34	75.34	71.49	78.70	67.27
CGDL [14]	91.79	77.42	70.02	78.52	72.7
GCM-CF (Ours)	**94.21**	**79.23**	**73.03**	**80.29**	**74.70**
Split	2				
Method	MNIST	SVHN	CIFAR10	C+10	C+50
Softmax	77.06	75.03	73.02	77.82	65.87
OpenMax [2]	86.97	77.27	73.74	79.02	67.56
CGDL [14]	86.76	73.63	73.15	76.46	70.79
GCM-CF (Ours)	**91.82**	**80.28**	**75.71**	**79.67**	**74.79**
Split	3				
Method	MNIST	SVHN	CIFAR10	C+10	C+50
Softmax	77.44	78.67	70.79	77.61	66.21
OpenMax [2]	83.39	80.00	72.01	78.38	67.83
CGDL [14]	92.36	77.59	74.77	77.92	71.93
GCM-CF (Ours)	**93.86**	**80.51**	**75.38**	**79.40**	**76.56**
Split	4				
Method	MNIST	SVHN	CIFAR10	C+10	C+50
Softmax	76.03	75.47	70.25	77.67	66.01
OpenMax [2]	87.06	76.80	70.76	78.64	68.21
CGDL [14]	90.34	73.53	70.30	78.27	71.69
GCM-CF (Ours)	**91.34**	**80.76**	**71.12**	**78.74**	**74.53**
Split	5				
Method	MNIST	SVHN	CIFAR10	C+10	C+50
Softmax	77.33	78.30	68.12	78.13	65.97
OpenMax [2]	89.88	80.33	68.90	78.70	67.55
CGDL [14]	83.51	79.41	66.89	78.00	68.18
GCM-CF (Ours)	**92.45**	**80.33**	**69.52**	**78.79**	**72.40**

Table A6: Comparison of the F1 score averaged over 5 random splits in OSR. Note that since the official code of CGDL [14] is not complete, we implemented the code of dataloader, F1 score and set part of parameters. Moreover, we also implemented Softmax and OpenMax [2] for evaluation. For GCM-CF, after binary classification, we used CGDL for supervised classification on the seen-classes.

Method	MNIST	SVHN	CIFAR10	C+10	C+50
Plain CNN	0.995	0.965	0.917	0.941	0.940
CGDL [14]	0.996	0.962	0.913	0.934	0.935

Table A7: Comparison of the Closed-Set accuracy in OSR.

Effect of disentanglement. To further demonstrate the effectiveness of disentangling Z and Y in OSR, we also compared an entangled model (without the counterfactual-faithful training) and the disentangled model on F1 scores. The results are shown in Table A8. Similar to the ZSL, we can also see the F1 scores of entangled model are much lower than those of disentangled model. The constructed green counterfactual samples are still closer to the unseen sample though given the seen attributes without disentanglement, which demonstrate the necessity of the proposed disentangle loss.

More Qualitative Results. Figure A1 and A2 show the additional qualitative results comparing existing

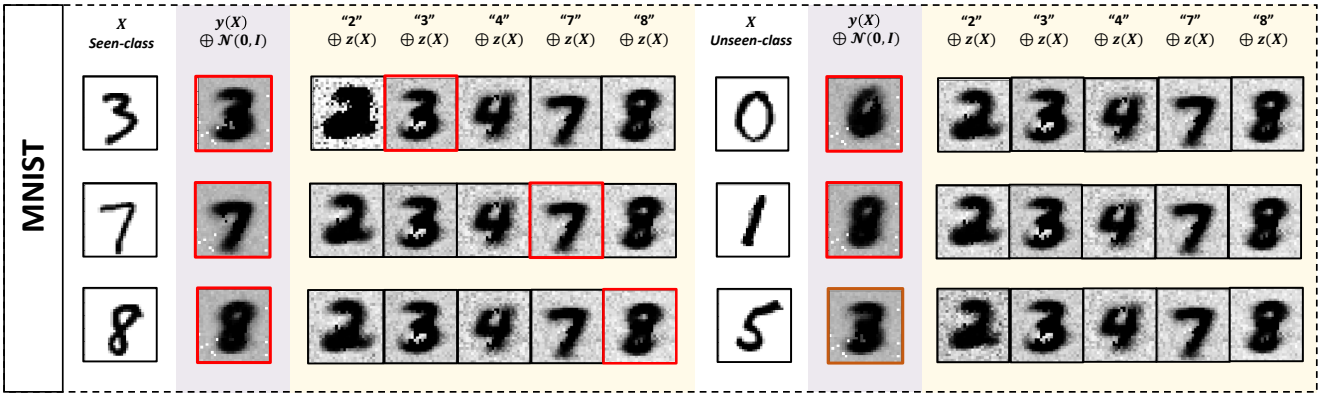

Figure A1: The additional qualitative results of the reconstructed images using CGDL [14] ($y(X) \oplus \mathcal{N}(0, I)$) and the counterfactual images generated from our GCM-CF ($y \oplus z(X)$) on MNIST dataset. The red box on a generated image denotes that it is similar to the original, while the brown box represents the failure generation.

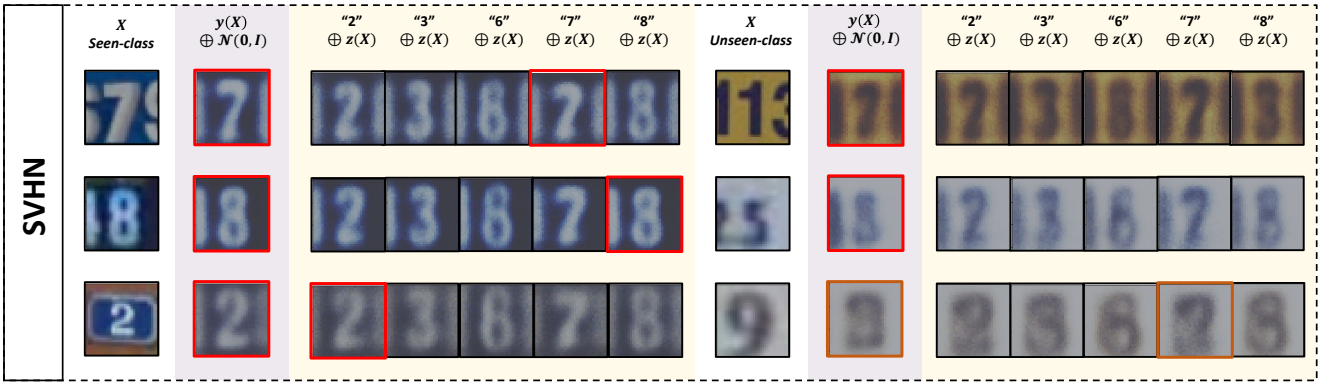

Figure A2: The additional qualitative results of the reconstructed images using CGDL [14] ($y(X) \oplus \mathcal{N}(0, I)$) and the counterfactual images generated from our GCM-CF ($y \oplus z(X)$) on SVHN dataset. The red box on a generated image denotes that it is similar to the original, while the brown box represents the failure generation.

Model	MNIST	SVHN	CIFAR10	C+10	C+50
Entangled	91.37	62.57	67.03	73.81	69.18
Disentangled	**94.21**	**79.23**	**73.03**	**80.29**	**74.70**

Table A8: Comparison of the F1 scores using entangeled and disentangled model in OSR.

reconstruction-based approach with our proposed counterfactual approach on MNIST and SVHN dataset. For the *Seen-class* (*i.e.*, the left part), both the baseline model and our GCM-CF can reconstruct samples with low reconstruction error, which means both of them can handle well given the seen-class images. When coming to the unseen-class (*i.e.*, the right part), the baseline method would still generate similar samples (red box), with a much lower reconstruction error comparing to the counterfactual samples produced by GCM-CF, resulting in a failure rejection to the unknown outlier. The brown box represents the failure reconstructions for the baseline model (*i.e.*, generated sample is also dissimilar with the original input image) given some unseen-class samples. Note that this is reasonable since the model haven't seen the unseen-class samples during training, which also corresponds to Figure 4b in the main paper. In this case, our counterfactual model can still make better generation (*e.g.*, "3" in the last row of Figure A1).

For the CIFAR10 dataset, as dicussed in the main paper, we cannot generate realistic images due to the conflict between visual perception and discriminative training. Therefore, we apply a pretrained image classifier to generate CAM to reveal the sensible yet discriminating features. Here we show the additional examples in Figure A3. The first row is the direct image reconstruction results generated by baseline and proposed model. The disordered appearance explains that the pixel-level generation is not sensible. However, when utilizing the tool of CAM, something magical happened. The insensible pixel-level generation becomes discriminative in the view of the pretrained classifier. The generation samples of our proposed GCM-CF reveal different heat maps given different counterfactual class conditions. Among them the heat map of "ship" condition is quite similar to that of the reconstruction of the baseline model (red box), showing the consistency of the CAM heat

map. Moreover, for different samples in the same class (*i.e.*, the different rows), the class-specific CAM heat maps keep stable with only minor changes. It further demonstrates that the CAM heat map can be considered a kind of substitution of the original pixel images to reveal the discrimitative feature.

References

[1] Mark Anthony Armstrong. *Basic topology*. Springer Science & Business Media, 2013. 1

[2] Abhijit Bendale and Terrance E Boult. Towards open set deep networks. In *CVPR*, 2016. 4

[3] M. Besserve, A. Mehrjou, R. Sun, and B. Schölkopf. Counterfactuals uncover the modular structure of deep generative models. In *ICLR*, 2020. 1

[4] Ali Farhadi, Ian Endres, Derek Hoiem, and David Forsyth. Describing objects by their attributes. In *CVPR*, 2009. 2, 3

[5] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 2

[6] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 3

[7] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. In *ICML*, 2016. 1

[8] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010. 3

[9] Jingjing Li, Mengmeng Jing, Ke Lu, Zhengming Ding, Lei Zhu, and Zi Huang. Leveraging the invariant side of generative zero-shot learning. In *CVPR*, 2019. 3

[10] Sanath Narayan, Akshita Gupta, Fahad Shahbaz Khan, Cees GM Snoek, and Ling Shao. Latent embedding feedback and discriminative features for zero-shot classification. In *ECCV*, 2020. 1, 2, 3

[11] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. In *NeurIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011. 3

[12] Michael Puthawala, Konik Kothari, Matti Lassas, Ivan Dokmanić, and Maarten de Hoop. Globally injective relu networks. *arXiv preprint arXiv:2006.08464*, 2020. 1

[13] Edgar Schonfeld, Sayna Ebrahimi, Samarth Sinha, Trevor Darrell, and Zeynep Akata. Generalized zero-and few-shot learning via aligned variational autoencoders. In *CVPR*, 2019. 2, 3

[14] Xin Sun, Zhenning Yang, Chi Zhang, Keck-Voon Ling, and Guohao Peng. Conditional gaussian distribution learning for open set recognition. In *CVPR*, 2020. 2, 3, 4, 5, 7

[15] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In *CVPR*, 2018. 3

[16] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD Birds 200. Technical report, California Institute of Technology, 2010. 2, 3

[17] Yongqin Xian, H. Christoph Lampert, Bernt Schiele, and Zeynep Akata. Zero-shot learning - a comprehensive evaluation of the good, the bad and the ugly. *TPAMI*, 2018. 2, 3

[18] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *CVPR*, 2010. 2, 3

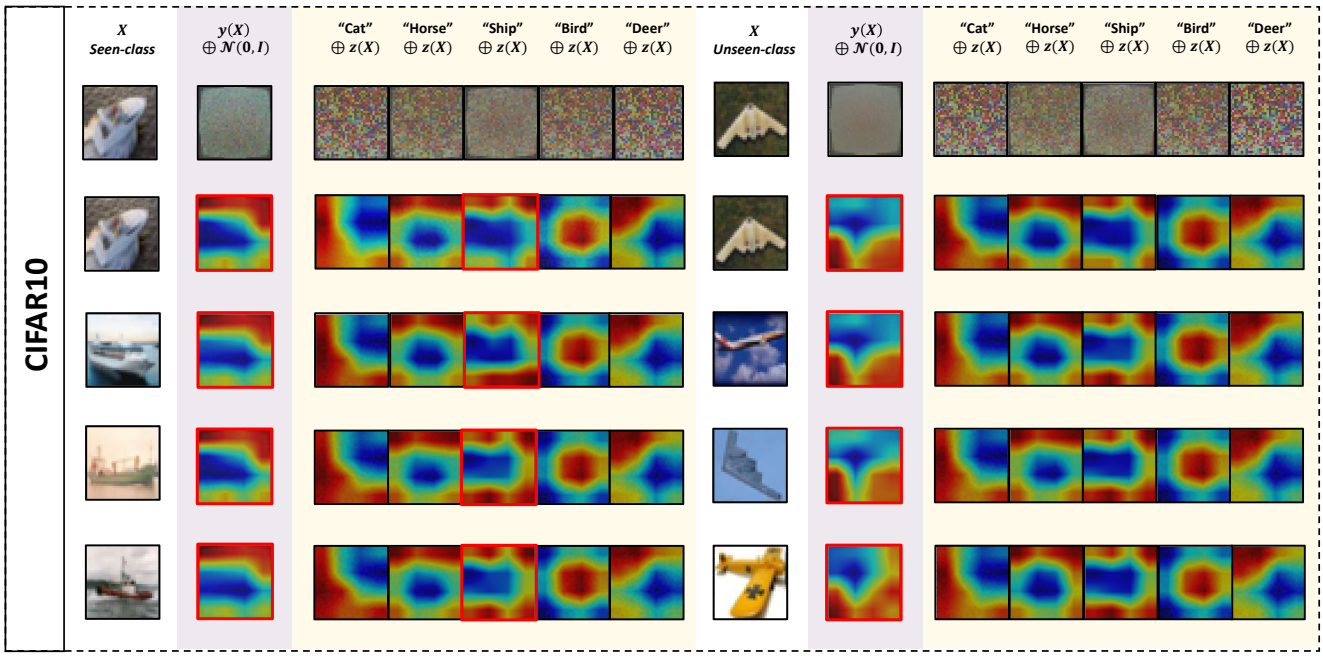

Figure A3: The additional qualitative results of the reconstructed images using CGDL [14] ($y(X) \oplus \mathcal{N}(0, I)$) and the counterfactual images generated from our GCM-CF ($y \oplus z(X)$) on CIFAR10 dataset. The red box on a generated image denotes that it is similar to the original.