CI-GAN : CO-CLUSTERING BY INFORMATION MAXIMIZING GENERATIVE ADVERSARIAL NETWORKS — APPENDIX

1. VARIATIONAL MUTUAL INFORMATION MAXIMIZATION FOR *AUTOENCODER*

1.1. InfoGAN

To understand the objective of *AutoEncoder*, we need to understand how InfoGAN exploits variational mutual information maximization [1] to overcome the difficulties in maximizing I of the following objective (H represents entropy).

$$\min_G \max_D V(D, G) - I(c; G(z, c))$$
$$I(c; G(z, c)) = H(c) - H(c|G(z, c))$$

Directly maximizing I is found to be difficult as the derivation involves a posterior distribution $P(c|\overline{x})$ where $\overline{x}$ represent the samples generated by $G(z, c)$. Therefore, Chen et al. exploit variational mutual information maximization and optimize the lower bound of $I(c; G(z, c))$, L_I, instead. The key idea is to use an auxiliary distribution $Q(c|\overline{x})$ to approximate $P(c|\overline{x})$ [2].

$$L_I(G, Q) = \mathbb{E}_{c\sim P(c), \overline{x}\sim G(z,c)}[\log Q(c|\overline{x})] + H(c)$$

Since the lower bound becomes tight as $Q(c|\overline{x})$ approaches the true distribution $P(c|\overline{x})$, the objective becomes the following.

$$\min_{G,Q} \max_D V(D, G) - L_I(G, Q)$$

1.2. *AutoEncoder*

As previously described in Section 3, we start from the following objective.

$$\min_{G,Q} \max_D V(D_r, G_r) + V(D_c, G_c) - I(x; Q(G(z, c)))$$
$$I(x; Q(G(z,c))) = I(x_r, x_c; Q_r(G_r(z_r, c_r)), Q_c(G_c(z_c, c_c)))$$

Though $I(x; Q(G(z, c)))$ look similar to $I(c; G(z, c))$ of InfoGAN, directly applying variational mutual information maximization to derive the lower bound is not possible due to the nested posterior distributions and the assumption that P_G has the same distribution as P_{data}. Therefore, we first derive $L_{\overline{x}}$, the lower bound for $I(\overline{x}; Q(G(z, c)))$, and exploit the similarity between $I(\overline{x}; Q(G(z, c)))$ and $I(x; Q(G(z, c)))$ to derive L_x, the lower bound for $I(x; Q(G(z, c)))$.

Before we dive into the detailed derivation, we accentuate *Lemma 5.1* which is originally introduced and proven by Chen et al. [2]. In this section, we replace x' with $\widehat{x}$ to be consistent with the notations used in this work.

Lemma 5.1 *For random variables X,Y and function $f(x, y)$ under suitable regularity conditions:* $\mathbb{E}_{x\sim X, y\sim Y|x}[f(x, y)] = \mathbb{E}_{x\sim X, y\sim Y|x, \widehat{x}\sim X|y}[f(\widehat{x}, y)]$

$L_{\overline{x}}$: Lower bound for $I(\overline{x}; Q(G(z, c)))$

Along with $P(c|\overline{x})$, the maximization of $I(\overline{x}; Q(G(z, c)))$ involves another posterior distribution $P(\overline{x}|c')$. Similar to how $Q(c|x)$ is used to approximate $P(c|x)$, we introduce another distribution $R(\overline{x}|c')$ for approximating $P(\overline{x}|c')$.

$$\begin{aligned}
I(\overline{x}; Q(G(z, c))) &= H(\overline{x}) - H(\overline{x}|Q(G(z, c))) && \text{(by the definition of } I) \\
&= H(\overline{x}) + \mathbb{E}_{c'\sim Q(G(z,c))}[\mathbb{E}_{\widehat{x}\sim P(\overline{x}|c')}[\log P(\widehat{x}|c')]] \\
&= H(\overline{x}) + \mathbb{E}_{c'\sim Q(\overline{x})}[\mathbb{E}_{\widehat{x}\sim P(\overline{x}|c')}[\log P(\widehat{x}|c')]] && (\overline{x} = G(z, c)) \\
&\geq H(\overline{x}) + \mathbb{E}_{c'\sim Q(\overline{x})}[\underbrace{D_{KL}(P(\cdot|c') \parallel R(\cdot|c'))}_{\geq 0} + \mathbb{E}_{\widehat{x}\sim P(\overline{x}|c')}[\log R(\widehat{x}|c')]]
\end{aligned}$$

$$
\begin{aligned}
&= H(\overline{x}) + \mathbb{E}_{c' \sim Q(\overline{x})}[\mathbb{E}_{\widehat{x} \sim P(\overline{x}|c')}[\log R(\widehat{x}|c')]] \\
&\hspace{20em} (R(\overline{x}|c') \text{ is an approximation for } P(\overline{x}|c')) \\
&= H(\overline{x}) + \mathbb{E}_{c' \sim Q(\overline{x}), \widehat{x} \sim P(\overline{x}|c')}[\log R(\widehat{x}|c')] \\
&= H(\overline{x}) + \mathbb{E}_{c' \sim P_Q(\overline{x}), \widehat{x} \sim P(\overline{x}|c')}[\log R(\widehat{x}|c')] \\
&= H(\overline{x}) + \mathbb{E}_{\overline{x} \sim P(\overline{x}), c' \sim P_Q(\overline{x}), \widehat{x} \sim P(\overline{x}|c')}[\log R(\widehat{x}|c')] \\
&= H(\overline{x}) + \mathbb{E}_{\overline{x} \sim P(\overline{x}), c' \sim P_Q(\overline{x})}[\log R(\overline{x}|c')] && \text{(by \textit{Lemma 5.1})} \\
&= H(\overline{x}) + \mathbb{E}_{\overline{x} \sim P(\overline{x}), c' \sim Q(\overline{x})}[\log R(\overline{x}|c')] \\
&= H(\overline{x}) + \mathbb{E}_{\overline{x} \sim P(\overline{x}), c' \sim Q(G(z,c))}[\log R(\overline{x}|c')] && (\overline{x} = G(z,c)) \\
&= L_{\overline{x}}(G, Q, R)
\end{aligned}
$$

Overall, as the distributions Q and R approach the true posterior distribution, the lower bound becomes tight increasing the mutual information $I(\overline{x}; Q(G(z,c)))$.

L_x : Lower bound for $I(x; Q(G(z,c)))$

To derive L_x, we first assume that P_G has the same distribution as P_{data} ($x = G(z,c)$). Also we leverage an auxiliary distribution $R(x|c')$ to approximate $P(x|c')$. Then, L_x can be derived in almost the same way.

$$
\begin{aligned}
I(x; Q(G(z,c))) &= H(x) - H(x|Q(G(z,c))) && \text{(by the definition of } I) \\
&= H(x) + \mathbb{E}_{c' \sim Q(G(z,c))}[\mathbb{E}_{\widehat{x} \sim P(x|c')}[\log P(\widehat{x}|c')]] \\
&= H(x) + \mathbb{E}_{c' \sim Q(x)}[\mathbb{E}_{\widehat{x} \sim P(x|c')}[\log P(\widehat{x}|c')]] && (x = G(z,c)) \\
&\geq H(x) + \mathbb{E}_{c' \sim Q(x)}[\underbrace{D_{KL}(P(\cdot|c') \parallel R(\cdot|c'))}_{\geq 0} + \mathbb{E}_{\widehat{x} \sim P(x|c')}[\log R(\widehat{x}|c')]] \\
&= H(x) + \mathbb{E}_{c' \sim Q(x)}[\mathbb{E}_{\widehat{x} \sim P(x|c')}[\log R(\widehat{x}|c')]] \\
&\hspace{20em} (R(x|c') \text{ is an approximation for } P(x|c')) \\
&= H(x) + \mathbb{E}_{c' \sim Q(x), \widehat{x} \sim P(x|c')}[\log R(\widehat{x}|c')] \\
&= H(x) + \mathbb{E}_{c' \sim P_Q(x), \widehat{x} \sim P(x|c')}[\log R(\widehat{x}|c')] \\
&= H(x) + \mathbb{E}_{x \sim P(x), c' \sim P_Q(x), \widehat{x} \sim P(x|c')}[\log R(\widehat{x}|c')] \\
&= H(x) + \mathbb{E}_{x \sim P(x), c' \sim P_Q(x)}[\log R(x|c')] && \text{(by \textit{Lemma 5.1})} \\
&= H(x) + \mathbb{E}_{x \sim P(x), c' \sim Q(x)}[\log R(x|c')] \\
&= H(x) + \mathbb{E}_{x \sim P(x), c' \sim Q(G(z,c))}[\log R(x|c')] && (x = G(z,c)) \\
&= L_x(G, Q, R)
\end{aligned}
$$

Given the two lower bounds L_x and $L_{\overline{x}}$, the differences can be summarized as follow:

1. L_x has a constant term $H(x)$ while $L_{\overline{x}}$ has $H(\overline{x})$
2. Generator for L_x has a distribution of P_{data} while the one for $L_{\overline{x}}$ has $P_{G(z,c)}$
3. The distribution R approximates $P(x|c')$ in the case of L_x while it approximates $P(\overline{x}|c')$ in the case of $L_{\overline{x}}$

Interestingly, the first difference can be ignored as it simply yields that the gap between the two lower bounds is bounded by a constant value. Furthermore, the second difference is minimized indirectly by GAN—throughout the training process, generator learns to produce realistic data; the distribution $P_{G(z,c)}$ becomes the distribution P_{data}. To minimize the difference between $P(x|c')$ and $P(\overline{x}|c')$, *AutoEncoder* exploits the network R which reconstructs the original pair of row and column samples from the two independent cluster labels c'_r and c'_c.

Altogether, the objective of *AutoEncoder* can be written as follows with the architecture in Figure 1.

$$\min_{G,Q} \max_{D} V(D_r, G_r) + V(D_c, G_c) - L_I(G_r, Q_r) - L_I(G_c, Q_c) - L_x(G, R)$$

2. MODEL ARCHITECTURE

As illustrated in Figure 1, *Combiner* and *AutoEncoder* share the same architectures for G, D and Q in order to ensure coherence. First of all, the size of z is set to 50 and the input data is assumed to have 196 features (14×14). The size of c depends on the dataset as it must be equal to the number of clusters.

Given a random variable z and a random encoding vector c, G first applies a fully connected (FC) layer and a batch normalization (BN) layer to generate a tensor of size $128 \times 7 \times 7$. We then upscale the tensor by factor of 2 and apply a 2D convolutional (CONV) layer reducing the number of channels to 64. With a stride of 1 for both dimensions, a padding of 1 on all sides, and a kernel of size 3×3, we then obtain a tensor of size $64 \times 14 \times 14$. Next, the tensor is fed into another BN layer with a rectified linear unit (ReLU) activation. The final layer of G is a 2D CONV layer with `tanh` activation.

D and Q share a single network for feature representation; we repeatably apply a group of layers which consists of a 2D CONV layer with ReLU activation, a dropout layer, and a BN layer. The stride of each CONV layer is set to 2 and the number of channels is reduced by factor of 2 starting from 16. Therefore, the final tensor has a size of $64 \times 2 \times 2$. D and Q then apply different FC layers to evaluate authenticity and to reconstruct the initial encoding vector c, respectively.

While *Combiner* has an additional FC layer for Q_{co} that reconstructs d, *AutoEncoder* has R which reconstructs the original pairs from encoding pairs. The network architecture of R is same as G except that their input and output tensor have different sizes; the input for R is the concatenation of c'_r and c'_c and the output is the concatenation of row and column samples.

3. HYPERPARAMETER TUNING FOR SYNTHETIC DATASETS

Since CI-GAN consists of multiple networks, we have applied grid search to find the best hyperparameter setting for each model. For *Combiner*, every component is trained with the learning rate of 0.00005. Q_{co} is trained once for every three epochs of row and column training. The quality of the generated co-clusters is found to be better when G is updated along with Q_{co}. On the other hand, *AutoEncoder* produces the best result with the learning rate of 0.0001. Training G along with R is found to be unnecessary when R is trained once for every five epochs of row and column training.

4. SAMPLE CO-CLUSTERS ON SYNTHETIC DATASETS

Figure 5∼9 are co-clusters generated from one of the experiments on synthetic datasets. Each entries are colored based on `gist_ncar_r` colormap of Matplotlib and the two dimensions are grouped based on their labels for better interpretation.

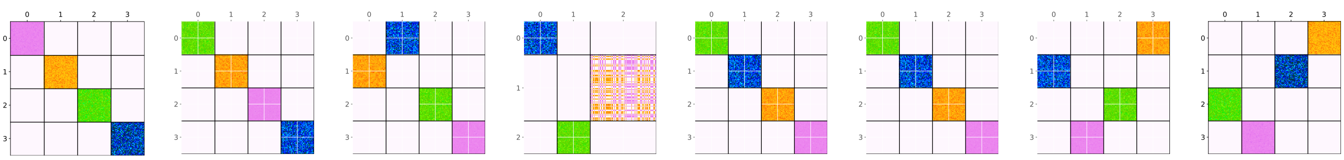

Fig. 5. Co-clusters and accuracy on `Diagonal` dataset. From left to right: Original, *Spec-Co* (1.000), *Spec-Bi* (1.000), *Info-CC* (0.579), *SA-CC* (1.000), *CoClus* (0.563), *Combiner* (1.000), *AutoEncoder* (1.000).

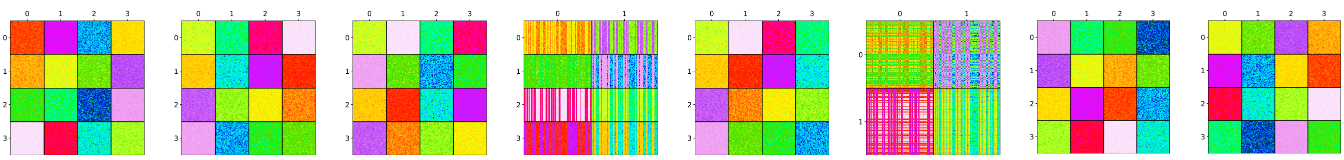

Fig. 6. Co-clusters and accuracy on `Checker-Shuffled` dataset. From left to right: Original, *Spec-Co* (1.000), *Spec-Bi* (1.000), *Info-CC* (0.636), *SA-CC* (1.000), *CoClus* (0.333), *Combiner* (1.000), *AutoEncoder* (1.000).

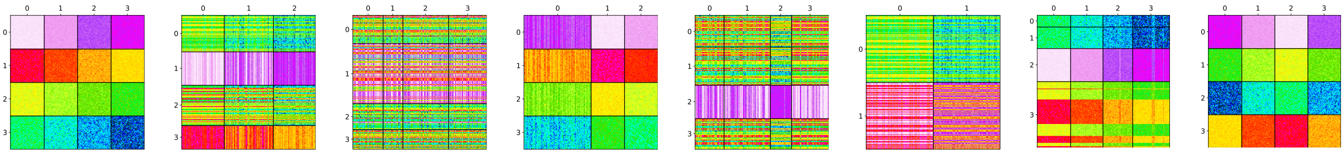

Fig. 7. Co-clusters and accuracy on `Checker-Ordered` dataset. From left to right: Original, *Spec-Co* (0.255), *Spec-Bi* (0.037), *Info-CC* (0.784), *SA-CC* (0.132), *CoClus* (0.333), *Combiner* (0.640), *AutoEncoder* (1.000).

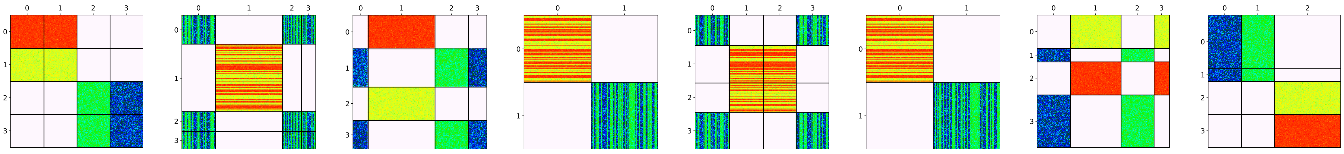

Fig. 8. Co-clusters and accuracy on `Mixed-Identical` dataset. From left to right: Original, *Spec-Co* (0.266), *Spec-Bi* (0.519), *Info-CC* (0.333), *SA-CC* (0.206), *CoClus* (0.333), *Combiner* (0.604), *AutoEncoder* (0.574).

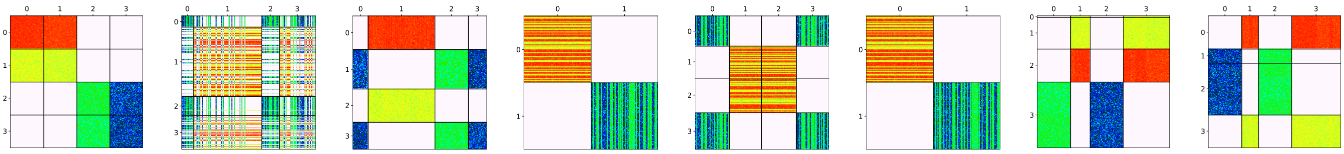

Fig. 9. Co-clusters and accuracy on `Mixed-Similar` dataset. From left to right: Original, *Spec-Co* (0.057), *Spec-Bi* (0.519), *Info-CC* (0.333), *SA-CC* (0.203), *CoClus* (0.333), *Combiner* (0.601), *AutoEncoder* (0.650).

5. REFERENCES

[1] D. Barber and F. Agakov, “The IM algorithm : A variational approach to information maximization,” *Advances in neural information processing systems*, vol. 16, 2004.

[2] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, “InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets,” in *Advances in neural information processing systems*, 2016.