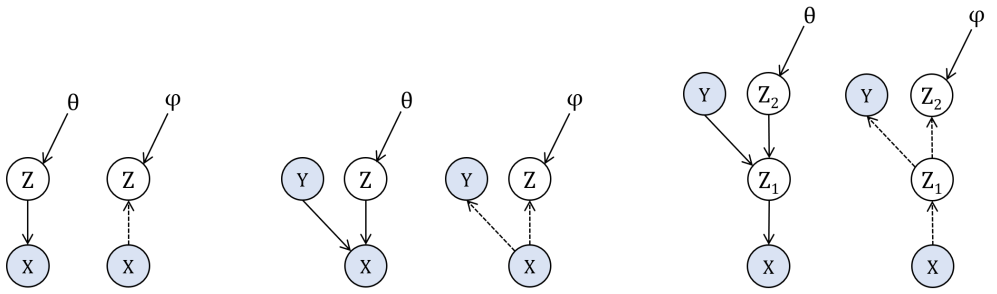
A Appendix

A.1 M1 and M2 Variational Auto-Encoders

As the first proposed model, the M1 VAE is the conventional model that is used to learn representations of data [Kingma and Welling, 2014, Rezende et al., 2014]. These features are learned from the covariate matrix X only. Figure 7(a) illustrates the encoder and decoder of the M1 VAE. Note the graphical model on the left depicts the encoder; and the one on the right depict the decoder, which has arrows going the other direction.

Proposed by Kingma et al. [2014], the M2 model was an attempt to incorporate the information in target Y into the representation learning procedure. This results in learning representations that separate specifications of individual targets from general properties shared between various targets. In case of digit generation, this translates into separating specifications that distinguish each digit from writing style or lighting condition. Figure 7(b) illustrates the encoder and decoder of the M2 VAE.

We can stack the M1 and M2 models as shown in Figure 7(c) to get the best results. This way, we can first learn a representation Z_1 from raw covariates, then find a second representation Z_2, now learning from Z_1 instead of the raw data.



(a) M1 model (b) M2 model (c) M1+M2 model

Figure 7: Decoders (parametrized by θ) and encoders (parametrized by ϕ) of the M1, M2, and M1+M2 VAEs.

A.2 Procedure of Generating the Synthetic Datasets

Given as input the sample size N; dimensionalities $[m_\Gamma, m_\Delta, m_\Upsilon] \in \mathcal{Z}^{+(3)}$; for each factor $L \in \{\Gamma, \Delta, \Upsilon\}$, the means and covariance matrices (μ_L, Σ_L); and a scalar ζ that determines the slope of the logistic curve.

- For each latent factor $L \in \{\Gamma, \Delta, \Upsilon\}$, form L by drawing N instances (each of size m_L) from $\mathcal{N}(\mu_L, \Sigma_L)$. The covariates matrix X is the result of concatenating Γ, Δ, and Υ. Refer to the concatenation of Γ and Δ as Ψ and that of Δ and Υas Φ (for later use).
- For treatment T, sample $m_\Gamma + m_\Delta$ tuple of coefficients θ from $\mathcal{N}(0,1)^{m_\Gamma + m_\Delta}$. Define the logging policy as $\pi_0(\, t=1 \,|\, z\,) \;=\; \frac{1}{1+\exp(-\zeta z)}$, where $z = \Psi \cdot \theta$. For each instance x_i, sample treatment t_i from the Bernoulli distribution with parameter $\pi_0(\, t=1 \,|\, z_i\,)$.
- For outcomes Y^0 and Y^1, sample $m_\Delta + m_\Upsilon$ tuple of coefficients ϑ^0 and ϑ^1 from $\mathcal{N}(0,1)^{m_\Delta + m_\Upsilon}$ Define $y^0 = (\Phi \circ \Phi \circ \Phi + 0.5) \cdot \vartheta^0 / (m_\Delta + m_\Upsilon) + \varepsilon$ and $y^1 = (\Phi \circ \Phi) \cdot \vartheta^1 / (m_\Delta + m_\Upsilon) + \varepsilon$, where ε is a white noise sampled from $\mathcal{N}(0, 0.1)$ and $\circ$ is the symbol for element-wise product.

A.3 Evaluating Identification of the Underlying Factors

Here, we elaborate on the procedure we followed to evaluate identification performance of the underlying factors. We produced four dummy vectors $V_i \in \mathbb{R}^{m_\Gamma + m_\Delta + m_\Upsilon + m_\Xi}$ as depicted on the left-side of Figure 8. The first to third vectors had ones (constant) in the positions associated with Γ, Δ, and Υ respectively, and the remainder of them were filled with zeroes. The fourth vector was all

ones, so we can measure the maximum amount of information that is passed to the final layer of each representation network.

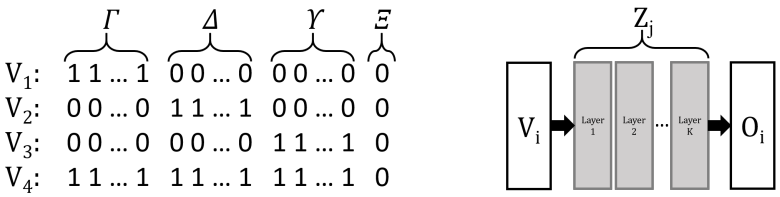


Figure 8: The four dummy x-like vectors (left); and the input/output vectors of the representation networks (right).

In the next step, each vector V_i is fed to each trained network Z_j, and the output O_i is recorded (see the right-side of Figure 8). The average of O_i represents the power of signal that was communicated from V_i and passed through the Z_j channel. The values reported in the tables illustrated in Figure 4 are the ratios of {average of O_1, O_2, O_3} divided by {average of O_4} for all the learned representation networks.

A.4 Hyperparameters

For all CFR, DR-CFR, and VAE-CI methods, we trained the neural networks with 3 layers (each consisting 200 hidden neurons)[8], non-linear activation function `elu`, regularization coefficient of λ=1E-4, `Adam` optimizer [Kingma and Ba, 2015] with a learning rate of 1E-3, batch size of 300, and maximum number of iterations of $10,000$. See Table 4 for our hyperparameter search space.

Table 4: Hyperparameters and ranges

Hyperparameter	**Range**
Discrepancy coefficient α	$\{0, 1E\{-3, -2, -1, 0, 1\}\}$
KLD coefficient β	$\{0, 1E\{-3, -2, -1, 0, 1, 2\}\}$
Generative coefficient γ	$\{0, 1E\{-5, -4, -3, -2, -1, 0\}\}$

A.5 A Detailed Analysis of the Effect of β

Our initial hypothesis in using β-VAE was that it might help *further* disentangle the underlying factors, in addition to the other constraint already in place (*i.e.*, the architecture as well as the discrepancy penalty). However, Figure 6(b) suggests that close-to-zero or even zero βs also work effectively. To further explore this hypothesis, we examined the decomposition tables (similar to Figure 4) of H-VAE-CI for extreme configurations with $\beta = 0$ and observed that they were all effective at decomposing the underlying factors Γ, Δ, and Υ (similar to the performance reported in the green table in Figure 4). Figure 9 shows several of these tables.

Our interpretation of this observation is that the H-VAE-CI's architecture already takes care of decomposing the Γ, Δ, and Υ factors, without needing the help of a KLD penalty. This means either of the following is happening: (i) β-VAE is not the best performing disentangling method and other disentangling constraints should be used instead — *e.g.*, works of Chen et al. [2018] and Lopez et al. [2018]; or (ii) it is theoretically impossible to achieve disentanglement without some supervision [Locatello et al., 2019], which might not be possible to provide in this task. Exploring these options is out of the scope of this paper and is left to future work.

[8] In addition to this basic configuration, we also perform our grid search with an updated number of layers and/or number of neurons in each layer. This makes sure that all methods enjoy a similar model complexity.

H-VAE-CI							
	Z1	Z2	Z3	Z4	Z5	Z6	Z7
Γ	0.2181	0.2185	**1.8791**	1.7711	0.2164	0.2190	0.2039
Δ	0.6041	0.6051	0.6142	0.6308	**0.8688**	0.8315	**0.6138**
ϒ	**0.8523**	0.8552	0.3321	0.3834	0.7552	0.7859	**0.7384**

H-VAE-CI							
	Z1	Z2	Z3	Z4	Z5	Z6	Z7
Γ	0.2242	0.2182	**0.7439**	0.6770	0.2373	0.2583	0.2189
Δ	0.3014	0.2963	0.2612	0.3169	**0.6051**	0.5845	**0.7048**
ϒ	**0.5385**	0.5430	0.3211	0.3303	0.4394	0.4412	**0.4571**

H-VAE-CI							
	Z1	Z2	Z3	Z4	Z5	Z6	Z7
Γ	0.4254	0.4493	**0.7090**	0.6872	0.3823	0.3771	0.3874
Δ	0.6438	0.6461	0.2750	0.3129	**0.7452**	0.7569	**0.8237**
ϒ	**0.7760**	1.1200	0.3137	0.3480	0.7240	0.7464	**0.6717**

H-VAE-CI							
	Z1	Z2	Z3	Z4	Z5	Z6	Z7
Γ	0.3646	0.3643	**1.1457**	0.8659	0.3942	0.4069	0.3166
Δ	0.5127	0.5307	0.6463	0.5794	**0.7016**	0.6717	**0.7652**
ϒ	**0.5565**	0.5780	0.4260	0.3964	0.4119	0.4234	**0.4534**

H-VAE-CI							
	Z1	Z2	Z3	Z4	Z5	Z6	Z7
Γ	0.8821	0.8752	**0.5805**	0.5782	0.3326	0.3309	0.4006
Δ	1.2542	1.2480	0.2843	0.3488	**0.8553**	0.8568	**0.9392**
ϒ	**1.914**	1.923	0.4498	0.4797	0.7791	0.7757	**0.7969**

H-VAE-CI							
	Z1	Z2	Z3	Z4	Z5	Z6	Z7
Γ	0.0850	0.0875	**1.8791**	1.7711	0.1464	0.1459	0.1833
Δ	0.6107	0.6085	0.6142	0.6308	**0.7851**	0.7937	**0.6878**
ϒ	**0.8349**	0.8242	0.3321	0.3834	0.5177	0.5073	**0.6832**

Figure 9: Decomposition tables for H-VAE-CI with $\beta = 0$.