Concept Bottleneck Generative Models

Aya Abdelsalam Ismail [*1] **Julius Adebayo** [*1] **Héctor Corrada Bravo** [1] **Stephen Ra** [1] **Kyunghyun Cho** [1 2 3]

Abstract

Despite their increasing prevalence, generative models remain opaque and difficult to steer reliably. To address these challenges, we present concept bottleneck (CB) generative models, a type of generative model where one of its internal layers—a concept bottleneck (CB) layer—is constrained to encode human-understandable features. While concept-botttleneck layers have been used to improve interpretability for supervised learning tasks, here we extend them generative models. The concept bottleneck layer partitions the generative model into three parts: the pre-concept bottleneck portion, the CB layer, and the post-concept bottleneck portion. To train CB generative models, we complement the traditional task-based loss function for training generative models with three additional loss terms: a concept loss, an orthogonality loss, and a concept sensitivity loss. The CB layer and these corresponding loss terms are model agnostic, which we demonstrate by applying them to three different families of generative models: generative adversarial networks, variational autoencoders, and diffusion models. On real-world datasets, across three types of generative models, steering a generative model with the CB layer outperforms several baselines.

1. Introduction

Improvements in generative modeling have led to these models being applied to produce photo realistic images (Saharia et al., 2022), video (Ho et al., 2022; Villegas et al., 2022), protein sequences (Ingraham et al., 2022), small molecules (De Cao and Kipf, 2018), and coherent text (Brown et al., 2020). However, current generative models admit little-to-no-room for interpretation and control of high-level properties. Consider a model trained to generate protein sequences; a domain expert might be interested in determining whether the model has captured desirable features like thermostability and toxicity. An important goal is then to use these features, as knobs, to steer the model to generate sequences that satisfy desired ranges of thermostability and toxicity. In this work, we seek generative models with *intrinsically interpretable components* that enable easier interpretation and steerability.

Challenges with interpreting and steering generative model representations. Current approaches for interpreting generative models cannot reliably indicate that a model's representations map to human-understandable features that the model relies on for its output. One approach (Belinkov, 2022) for interpreting a generative model's representations uses a low-complexity model to predict a human understandable feature from the model's representations; however, high predictive performance of the low-complexity model does not mean the model's output is sensitive or reliant on that feature (Lovering and Pavlick, 2022). A different approach constrains the model to learn *disentangled* (Higgins et al., 2017; Tran et al., 2017; Meo et al., 2023) representations—that is, representations that can be decomposed into independent factors; nevertheless, it is generally not possible to guarantee that a disentangled representation is human interpretable (Locatello et al., 2019). Other approaches project model representations into lower dimensions and then search for directions correlated with human interpretable features (Härkönen et al., 2020); yet, since the model was not constrained to learn such features by design, it is possible that its representations do not encode the desired features.

Concept bottleneck generative models. To address challenges with current approaches for interpreting and steering current generative models, we present concept bottleneck (CB) generative models, a generative model where one of its internal layers—a concept bottleneck (CB) layer—is constrained to encode human-understandable features. We insert the CB layer into the generative model's architecture to give three parts: the pre-concept bottleneck portion, the CB layer, and the post-concept bottleneck portion. The pre-concept bottleneck portion maps from the input to activations, which are then mapped into human understandable features by the CB layer. The pre-defined concepts alone can be incomplete, so we allow additional representational

[*]Equal contribution [1]Genentech [2]Department of Computer Science, New York University [3]Center for Data Science, New York University. Correspondence to: Aya Abdelsalam Ismail <ismail.aya@gene.com>.

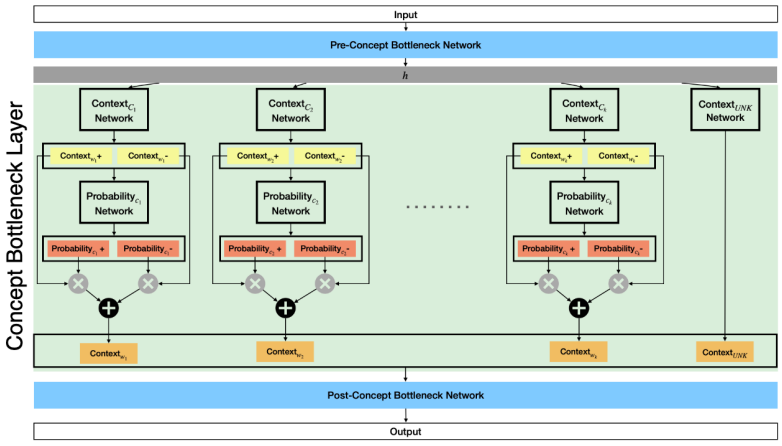


Figure 1: Concept Bottleneck Layer

capacity for unknown concepts, which are constrained to be orthogonal to the pre-defined features. Lastly, the post-concept bottleneck layer maps both the output of the CB layer and unknown concepts into a generated output. The CB layer is model agnostic, which we demonstrate by applying them to three different families of generative models: generative adversarial networks (Goodfellow, 2016; Radford et al., 2015; Creswell et al., 2018), variational autoencoders (Kingma and Welling, 2013; Kingma et al., 2019), and diffusion models (Sohl-Dickstein et al., 2015; Song et al., 2020; Ho et al., 2020). Using the CB layer, we are able to demonstrate the following capabilities:

Steering Generative Models: By intervening on the output of the CB layer, we can modulate the level of a particular concept that is present in the output of a generative model. We leverage this capability to control single concepts independently, and multiple concepts.

Understanding and Debugging Generative Models: The CB layer can also be used to debug a generative model during and post training. In Section 4.2, we show that the CB layer can help identify the important concepts that are responsible for a model's generated output. Similarly, the output of the concept bottleneck (CB) layer can be used to distinguish a model that has learned the pre-defined human-understandable features from a model that hasn't.

2. Setup & Background

In this section, we give an overview of concept bottleneck models, and the three types of generative models that we consider. We assume that all training samples come with pre-defined human understandable features, for example, CelebFaces Attributes dataset (Liu et al., 2015) that comes annotated with 40 concepts (e.g., male, smiling..etc), as such: $\{(x_i, c_i)\}_{i=1}^n$, where an example $x_i \in \mathbb{R}^d$ and an associated concept $c_i \in \mathbb{R}^k$ with $k << d$.

Generative models consider the task of modeling the probability distribution of an observed random variable $\mathbf{x} \sim \mathbb{P}(\mathbf{x})$, using a chosen model family $p_\theta(\mathbf{x})$ where θ is the parameters of the selected model family. We will refer to the standard objective for each type of generative model family as the task loss: $\mathcal{L}_{\text{task}}$. We consider 3 generative model families: VAEs (Kingma and Welling, 2013), GANs (Goodfellow et al., 2014), and diffusion models (Ho et al., 2020).

Concept Bottleneck Models & Concept Embedding Models. Concept Bottleneck Models (Koh et al., 2020) (CBMs) aim to replace black-box DNNs with interpretable models by first learning to predict a set of concepts, that is, 'interpretable' (e.g., hair color, gender), and then using these concepts to learn a downstream classification task. CBMs map samples x_i to labels y_i by first mapping x_i to an intermediate representation $c_i = h(x_i)$, where c_i are the understandable human concepts. An 'interpretable' label predictor, f, then maps the predicted concepts to labels: $y = f\left(h\left(x\right)\right)$. Consequently, the goal is to learn the following relationship $x_i \xrightarrow{h} c_i \xrightarrow{f} y_i$. The functions h and f can be learned jointly, sequentially or independently. However, CBMs often result in a lower task accuracy, especially when concept set is insufficient to characterize the label. Espinosa Zarlenga et al. (2022) proposed Concept Embedding Models (CEMs) to address this accuracy-interpretability trade-off. Instead of a scalar, CEMs map the input to a high-dimensional representation for each concept, and have predictive performance that is competitive with unconstrained models.

Variational Autoencoder. VAEs (Kingma and Welling, 2013; Kingma et al., 2019) consist of two components: an encoder and a decoder. The encoder, $q(z_i|x_i)$, maps an input, x_i, to a distribution over the latent variable, z_i. The output of the encoder parameterizes a Gaussian density, from which we sample a latent vector z_i^d. VAEs regularizes

this latent distribution to be similar to the prior distribution $p(z)$ which is typically $z \sim \mathcal{N}(0, I)$. The vector z_i^d is passed to the decoder, $p(x_i|z_i^d)$, a function that maps the latent vector to a distribution over the input. The task loss for the VAE is the negative log-likelihood:

$$\mathcal{L}(x_i)_{\text{task,vae}} = D_{KL}\big(q\left(z_i|x_i\right)||p\left(z_i\right)\big) - \mathbb{E}\big[\log p(x_i|z_i^d)\big], \tag{1}$$

where D_{KL} is the Kullback-Leibler divergence.

Generative Adversarial Networks. GANs (Goodfellow et al., 2014) consist of two components: a generator G that captures the data distribution and a discriminator D that estimates the probability that a sample came from the training data rather than G; both models are trained simultaneously. The generator learns to map a noise vector z_i to an output $\hat{x_i} = G(z_i)$. Conditional GANs (Mirza and Osindero, 2014b; Chen et al., 2016; Odena et al., 2017) augment the input to the generator with the concepts and also learn $G(z_i|c_i)$. They however do not learn $c_i = h(z_i)$. To learn $c_i = h(z_i)$, we first encode x_i into a latent vector $q(z_i|x_i)$, since $p(c_i|x_i)$ is known, i.e., concepts for a given sample are known, we now can learn $c_i = h(z_i)$. Similar to VAEs, we then sample a new latent vector z_i^d and use this for generation $G(z_i^d)$. This approach has been widely employed (Larsen et al., 2016; Isola et al., 2017; Wang et al., 2018) for image generation. For training, we use the loss introduced by VAE-GANs(Larsen et al., 2016), which combines the VAE encoder regularization prior loss with a GAN loss.

$$\mathcal{L}(x_i)_{\text{task,gan}} = D_{KL}\big(q\left(z_i|x_i\right)||p\left(z_i\right)\big) + \mathbb{E}\big[\log D\left(x_i\right)\big] + \mathbb{E}\big[1 - \log D\left(G\left(z_i^d\right)\right)\big]. \tag{2}$$

Diffusion models. Diffusion models can be interpreted as latent variable models with two stages (Sohl-Dickstein et al., 2015; Ho et al., 2020) . Given input data x_i, the first stage is the forward diffusion process, which involves incrementally adding Gaussian noise to the input and can be described as: $q\left(x_i^t|x_i^{t-1}\right) = \mathcal{N}\left(x_i^t; \sqrt{1-\beta_t}x_i^t, \beta_t I\right)$, where β_t is determined according to a pre-specified schedule. The second stage of the process learns a denoising model, $p\left(x_i^t|x_i^{t-1}\right)$ to reverse the forward process. The diffusion model is trained to maximize the lower bound to the marginal likelihood of the data, which can be reformulated into a mean-squared error loss as:

$$\mathcal{L}(x_i^t)_{\text{task,df}} = \sum_{i=1}^{T} \mathbb{E}\big\|\mu(x_i^t, t) - \hat{\mu}\left(x_i^t, x_i^{t=0}\right)\big\|^2. \tag{3}$$

We refer to (Ho et al., 2020; Weng, 2021; Rogge and Rasul, 2022) for a more detailed overview of diffusion models.

3. Concept Bottleneck Generative Models

We propose to insert a concept embedding layer—which we term a concept bottleneck (CB) layer—into a generative model. Our overall framework, shown in Figure 1, consists of 3 parts: the portion of the generative model before the CB layer (the pre-concept bottleneck network); the CB layer, and the portion of the generative model after the CB layer (the post-concept bottleneck network). The pre-concept bottleneck and post-concept bottleneck networks are specific to the family of models (GANs, diffusion, & VAE) used for generation, while the CB is common across all generative model families. We denote the pre-bottleneck network as e, its output as h, the CB layer as f, and the post-bottleneck network as g. In this section, we introduce the architecture of the CB layer, the constituent loss functions, and discuss how to intervene on the CB layer.

3.1. Architecture

We adapt the CEM layer of Espinosa Zarlenga et al. (2022) to the generative model setting. However, unlike in supervised learning tasks where the concept set is assumed to be near complete (Koh et al., 2020; Espinosa Zarlenga et al., 2022; Yuksekgonul et al., 2022), it is unrealistic to expect the pre-defined human understandable features—concepts—to be complete in the generative setting. For example, consider the task of generating human face; finding a comprehensive set of concepts that can control every generation aspect is challenging. However, one might have available concepts such as hair color, eye color, and skin tone amongst other attributes. We extend the CEM layer of Espinosa Zarlenga et al. (2022) to also account for unknown concepts that might also be required for generation. The proposed modification of the concept bottleneck layer is shown in Figure 1, and consists of k concept networks and an extra unknown-concept network (similar to (Deng et al., 2020; Shoshan et al., 2021)).

Each concept is represented with two embeddings: $w_i^+, w_i^- \in \mathbb{R}^m$ representing the active and inactive concept states, respectively. The output of the pre-concept bottleneck portion, embedding vector h, is fed into a context network ϕ that maps h into two embeddings per concept. This context network can be viewed as two separate functions: $w_i^+ = \phi^+(h)$ and $w_i^- = \phi^-(h)$. Embeddings w_i^+ and w_i^- are encouraged to be aligned with ground-truth concept c_i by the function Ψ_i trained to predict the probability of concept c_i being active from the joint embedding space, $\hat{c}_i = \Psi_i([w_i^+, w_i^-]^T) \in [0, 1]$. The final context embedding w_i is then constructed as a weighted mixture of the two embedding $w_i = \left(\hat{c}_i w_i^+ + (1 - \hat{c}_i)\, w_i^-\right)$. We refer to the Appendix on how to extend to continuous concepts. These concept embeddings from all k concepts are then concatenated together with the non-concept embed-

dings $w = [w_1, w_2, \ldots, w_{k+1}]$, resulting in a bottleneck $f(h) = w$ with size $m(k+1)$, which is fed into the post-concept network. Concept embeddings are interpretable, since we can interpret the degree to which the concept applies to a generated instance by observing the concept probability. For details on how we add a CB layer to different models refer to Appendix A

Intervention on Concept. CBGMs support test-time interventions, which allows the user to steer the output of the generative model. To intervene on concept c_i, one simply replaces the probability of the concept being active, $\hat{c}_i$, with the desired probability $\bar{c}_i$. The CB layers' context embedding, $w_i = \left(\bar{c}_i w_i^+ + (1 - \bar{c}_i)\, w_i^-\right)$, is combined as a convex combination of the positive and negative context vectors. The new context embedding will then be passed to the post-concept bottleneck model to generate the new output with the desired concept.

3.2. Loss Functions & Training

We train CB generative models in an end-to-end fashion by jointly minimizing the following augmented loss function: $\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{task}} + \alpha\mathcal{L}_{\text{con}} + \beta\mathcal{L}_{\text{orth}} + \gamma\mathcal{L}_{\text{sens}}$,

where $\mathcal{L}_{\text{task}}$ is the task loss, $\mathcal{L}_{\text{con}}$ is the concept loss, $\mathcal{L}_{\text{orth}}$ is the concept orthogonality loss and, $\mathcal{L}_{\text{sens}}$ is the concept sensitivity loss. The hyperparameters α, β and γ control the relative importance of the concept, orthogonality, sensitivity, and task losses. While the task loss is specific to each generative model family, $\mathcal{L}_{\text{con}}$, $\mathcal{L}_{\text{orth}}$ and $\mathcal{L}_{\text{sens}}$ are common across different models.

- **Task Loss**: the task loss is the traditional objective function for each generative model family as defined in Equations 1, 2, and 3, respectively.

- **Concept Loss**: across all generative model classes, the concept loss, $\mathcal{L}_{\text{con}}$, is the binary cross-entropy loss on the output of each probability network of the CB layer.

- **Concept Orthogonality Loss**: Since the concept set is incomplete, we allow for additional representation capacity for unknown concepts. However, these unknown concepts can also turn out to be transformations of the known concepts, which is undesirable. To prevent this, we encourage the unknown concepts to be orthogonal to the outputs of each concept network using an orthogonality constraint Ranasinghe et al. (2021) by minimizing the cosine similarity between the concept context embedding and the unknown context embedding within a mini-batch as follows:

$$\mathcal{L}_{\text{orth}} = \sum_{j \in B} \frac{\sum_{i=1}^{i=k} \left|\langle w_i\,,\, w_{k+1}\rangle\right|}{\sum_{i=1}^{i=k} 1} \tag{4}$$

where $\langle \cdot\,,\, \cdot \rangle$ is the cosine similarity applied to two embedding, $|\cdot|$ is the absolute value, and B denotes mini-batch size. The cosine similarity in the above equation involves the normalization of features such that $\langle x_i\,,\, x_j\rangle = \frac{x_i \cdot x_j}{\|x_i\|_2 \cdot \|x_j\|_2}$, where $\|\cdot\|_2$ is l_2 norm.

- **Concept Sensitivity Loss**: An undesirable scenario might occur where the post-bottleneck portion of the generative model bases sole generation on the unknown concept vectors. To avoid this, we encourage the post-bottleneck portion of the generative model to strongly depend the concept vectors via the concept sensitivity loss. We achieve this by intervening on the concept vector as described in section 3.1. The original pipeline is shown in Equation 5a. We intervene by replacing $\hat{c}$ with $\bar{c}$, changing the output of the CB layer and the generative model as shown in Equation 5b. The generated output is passed as an input to the network Equation 5c. We minimize the binary cross-entropy loss between the known concepts $\bar{c}$ and predicted concepts $\hat{c}$. This forces the generator to rely on the concept vectors.

$$x_i \xrightarrow{e} h_i \rightarrow \overbrace{\left(\left[w^+, w^-\right] \rightarrow \hat{c} \rightarrow w\right)}^{\text{CB layer}} \xrightarrow{g} y_i \tag{5a}$$

$$x_i \xrightarrow{e} h_i \rightarrow \left(\left[w^+, w^-\right] \rightarrow \bar{c} \rightarrow w\right) \xrightarrow{g} y_i \tag{5b}$$

$$y_i \xrightarrow{e} h_i \rightarrow \left(\left[w^+, w^-\right] \rightarrow \hat{c} \rightarrow w\right) \xrightarrow{g} y_i \tag{5c}$$

Taken together, we now have a full scheme for instantiating a concept-bottleneck generative model for any of the three widely-used types of generative models.

4. Experiments & Results

In this section, we answer the following questions: 1) **Steerability**: How can the CB layer be used to control the output of the generative model? 2) **Interpretability**: How can the CB layer be used to help interpret and debug a generative model? Additional experiments investigating the effect of CB layer on generation quality, details about different datasets, and baselines are available in Appendix B.

4.1. Steering CB Generative Models

Experimental Setup. We train a classifier for each concept. For each model, we generate 1000 samples where all classifiers return false, i.e., the concept was not detected in the generated image. Given the generated images with no concepts detected, we consider intervening on a single concept at a time by switching that concept 'on' (switching a concept 'on' is method dependent; for example, in

CGAN, this involves appending the value of the concept to the input, while for CBGMs, we intervene by changing the probabilities generated in the concept vector as described previously). We show both quantitative and qualitative results for both single-concept steerability on the Celeb-A dataset. Multi-concept steerability experiments are available in B. For empirical evaluation of concept steerability, we calculate the accuracy of a concept classifier when that concept is turned 'on'.

Results. Figure 2 show images generated by CB-GAN before and after single concept intervention. We find that by changing the concept probability vector we can control the generated output. We report the single-concept steerability metric in Table 1. At a high level, for each class of generative model and across most features considered, we find that controlling the presence of a concept in the generated output is more effective with concept-bottleneck generative models than current approaches. In GANs, for prominent concepts like gender, steering the model output with the CB layer can be five times as effective as compared to the closest baseline (ACGAN). In the diffusion model setting, the concept-bottleneck diffusion model outperforms classifier-free diffusion control across all 5 attributes considered. Similar results were also found in VAEs.

Class	High Cheekbones	Male	Mouth Open	Smiling	Wavy Hair
CGAN	5.8 ± 0.9	6.0 ± 0.6	6.1 ± 0.9	3.6± 0.9	13.5±1.47
ACGAN	11.8± 1.4	9.3± 0.8	13.5± 0.9	14.3±1.1	8.4 ± 0.9
InfoGAN	7.6 ± 0.7	6.6 ± 0.5	6.1± 0.7	6.8 ± 0.8	8.7 ± 0.7
CB-GAN	9.8± 0.9	53.7± 1.6	8.2± 0.6	25.8± 1.8	30.5± 1.6
CF-Diffusion	8.3±4.1	10.2±5.5	7.2±4.6	7.1±5.4	3.8±1.7
CB Diffusion	11.7±1.1	14.8±2.4	13.9±1.3	15.1±3.6	10.3±2.5
CVAE	4.8±2.4	3.5±1.3	3.9±1.9	8.9±2.0	9.1±1.5
CB-VAE	12.5±1.4	14.3±3.1	14.2±2.6	19.4±4.5	15.7±2.4

Table 1: Single concept steering.

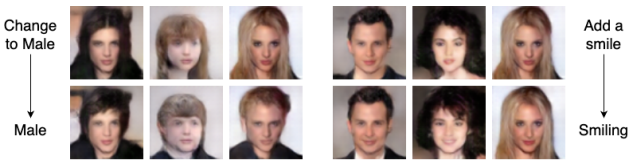


Figure 2: Single-concept interventions on CB-GAN.

4.2. Interpretability

A longstanding challenge with current generative models is that it is difficult to gain insight into what features are key for the output of the generative model. In this section, we show how the concept layer can be used for debugging a generative model during training time and assessing the quality of a generative model during inference. Additional experiments on Interpreting the generated output are available in Appendix B.

4.2.1. Model Debugging

Experimental Setup. Here, we use the CB layer to debug a generative model during training and inference. We train two CB-GAN models on color MNIST: (a) Model-a is trained on the ground truth set of concepts; (b) Model-b we corrupt one concept (red concept) by randomly flipping the label. During training, we track the concept loss and the concept validation accuracy on a test dataset. We then use the fully-trained models to generate 1000 random samples; we plot the probability histogram for each concept and use this to measure the ability of each model to capture the concept distribution.

Training-time debugging. Figure 3-a shows that both models were able to learn the green concept, i.e., both models have high accuracy on the validation dataset. Figure 3-b shows that Model-a was able to learn the red concept, while corrupted Model-b failed to learn red. By inserting a CB layer, we were able to track the model's ability to learn the desired properties while training.

Test-time debugging. Figure 3-c shows that concept probabilities for Model-a are centered around 1 (concept is on) and 0 (concept is off); hence Model-a was able to capture the data distribution for both concepts. In contrast, the red concept distribution for Model-b was centered around 0.5, which shows that Model-b could not learn the red concept's data distribution.

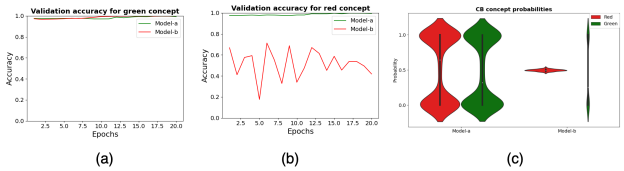


Figure 3: (a) Both have high accuracy for green. (b) Model-a has high accuracy for the red concept, but Model-b's accuracy was random. (c) Model-a captured the data distribution; Model-b could not learn the red concept's data distribution.

5. Conclusion

Due to unprecedented improvements, there has been an increased use of generative models across various settings. However, these models are mostly inscrutable and difficult to steer. In this paper, we present concept bottleneck generative models, a type of generative model where one of its internal layers—a concept bottleneck (CB) layer—is constrained to map from input representations to human-understandable features. The CB layer can be used as a simple plug-in module across different types of generative models. We show that inserting the CB layer does not hurt generation quality but helps to better steer and debug the models during and post-training. Overall, we see this work as a stepping stone for new kinds of generative models that are easier to understand and debug.

References

Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019.

David Bau, Steven Liu, Tongzhou Wang, Jun-Yan Zhu, and Antonio Torralba. Rewriting a deep generative model. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16*, pages 351–369. Springer, 2020.

Yonatan Belinkov. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*, 48(1):207–219, 2022.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. *Advances in neural information processing systems*, 29, 2016.

Zhi Chen, Yijie Bei, and Cynthia Rudin. Concept whitening for interpretable image recognition. *Nature Machine Intelligence*, 2(12):772–782, 2020.

Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A Bharath. Generative adversarial networks: An overview. *IEEE signal processing magazine*, 35(1):53–65, 2018.

Nicola De Cao and Thomas Kipf. Molgan: An implicit generative model for small molecular graphs. *arXiv preprint arXiv:1805.11973*, 2018.

Yu Deng, Jiaolong Yang, Dong Chen, Fang Wen, and Xin Tong. Disentangled and controllable face image generation via 3d imitative-contrastive learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5154–5163, 2020.

Mateo Espinosa Zarlenga, Pietro Barbiero, Gabriele Ciravegna, Giuseppe Marra, Francesco Giannini, Michelangelo Diligenti, Zohreh Shams, Frederic Precioso, Stefano Melacci, Adrian Weller, et al. Concept embedding models: Beyond the accuracy-explainability trade-off. *Advances in Neural Information Processing Systems*, 35:21400–21413, 2022.

Ian Goodfellow. Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

Erik Härkönen, Aaron Hertzmann, Jaakko Lehtinen, and Sylvain Paris. Ganspace: Discovering interpretable gan controls. *Advances in Neural Information Processing Systems*, 33:9841–9850, 2020.

Peter Hase, Mohit Bansal, Been Kim, and Asma Ghandeharioun. Does localization inform editing? surprising differences in causality-based localization vs. knowledge editing in language models. *arXiv preprint arXiv:2301.04213*, 2023.

Marton Havasi, Sonali Parbhoo, and Finale Doshi-Velez. Addressing leakage in concept bottleneck models. In *Advances in Neural Information Processing Systems*, 2022.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 2017.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International conference on learning representations*, 2017.

Irina Higgins, David Amos, David Pfau, Sebastien Racaniere, Loic Matthey, Danilo Rezende, and Alexander Lerchner. Towards a definition of disentangled representations. *arXiv preprint arXiv:1812.02230*, 2018.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey Gritsenko, Diederik P Kingma, Ben Poole, Mohammad Norouzi, David J Fleet, et al. Imagen video: High definition video generation with diffusion models. *arXiv preprint arXiv:2210.02303*, 2022.

John Ingraham, Max Baranov, Zak Costello, Vincent Frappier, Ahmed Ismail, Shan Tie, Wujie Wang, Vincent Xue, Fritz Obermeyer, Andrew Beam, et al. Illuminating protein space with a programmable generative model. *bioRxiv*, pages 2022–12, 2022.

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, et al. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In *International conference on machine learning*, pages 2668–2677. PMLR, 2018.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Diederik P Kingma, Max Welling, et al. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 2019.

Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. Concept bottleneck models. In *International Conference on Machine Learning*, pages 5338–5348. PMLR, 2020.

Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. In *International conference on machine learning*. PMLR, 2016.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

Francesco Locatello, Stefan Bauer, Mario Lucic, Gunnar Raetsch, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. Challenging common assumptions in the unsupervised learning of disentangled representations. In *international conference on machine learning*, pages 4114–4124. PMLR, 2019.

Max Losch, Mario Fritz, and Bernt Schiele. Interpretability beyond classification output: Semantic bottleneck networks. *arXiv preprint arXiv:1907.10882*, 2019.

Charles Lovering and Ellie Pavlick. Unit testing for concepts in neural networks. *Transactions of the Association for Computational Linguistics*, 10:1193–1208, 2022.

Anita Mahinpei, Justin Clark, Isaac Lage, Finale Doshi-Velez, and Weiwei Pan. Promises and pitfalls of black-box concept learning models. *arXiv preprint arXiv:2106.13314*, 2021.

Andrei Margeloiu, Matthew Ashman, Umang Bhatt, Yanzhi Chen, Mateja Jamnik, and Adrian Weller. Do concept bottleneck models learn as intended? *arXiv preprint arXiv:2105.04289*, 2021.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. *Advances in Neural Information Processing Systems*, 35:17359–17372, 2022.

Cristian Meo, Anirudh Goyal, and Justin Dauwels. Tc-vae: Uncovering out-of-distribution data generative factors. *arXiv preprint arXiv:2304.04103*, 2023.

Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014a.

Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014b.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. Fast model editing at scale. *arXiv preprint arXiv:2110.11309*, 2021.

Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. In *International conference on machine learning*, pages 2642–2651. PMLR, 2017.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

Kanchana Ranasinghe, Muzammal Naseer, Munawar Hayat, Salman Khan, and Fahad Shahbaz Khan. Orthogonal projection loss. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12333–12343, 2021.

Vikas Raunak and Arul Menezes. Rank-one editing of encoder-decoder models. *arXiv preprint arXiv:2211.13317*, 2022.

Niels Rogge and Kashif Rasul. The annotated diffusion model. June 2022. URL https://huggingface.co/blog/annotated-diffusion.

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in Neural Information Processing Systems*, 35:36479–36494, 2022.

Shibani Santurkar, Dimitris Tsipras, Mahalaxmi Elango, David Bau, Antonio Torralba, and Aleksander Madry. Editing a classifier by rewriting its prediction rules. *Advances in Neural Information Processing Systems*, 34:23359–23373, 2021.

Maximilian Seitzer. pytorch-fid: FID Score for PyTorch. https://github.com/mseitzer/pytorch-fid, August 2020. Version 0.3.0.

Alon Shoshan, Nadav Bhonker, Igor Kviatkovsky, and Gerard Medioni. Gan-control: Explicitly controllable gans. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 14083–14093, 2021.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265. PMLR, 2015.

Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. *Advances in neural information processing systems*, 28, 2015.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

Luan Tran, Xi Yin, and Xiaoming Liu. Disentangled representation learning gan for pose-invariant face recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1415–1424, 2017.

Ruben Villegas, Mohammad Babaeizadeh, Pieter-Jan Kindermans, Hernan Moraldo, Han Zhang, Mohammad Taghi Saffar, Santiago Castro, Julius Kunze, and Dumitru Erhan. Phenaki: Variable length video generation from open domain textual description. *arXiv preprint arXiv:2210.02399*, 2022.

Sheng-Yu Wang, David Bau, and Jun-Yan Zhu. Sketch your own gan. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14050–14060, 2021.

Sheng-Yu Wang, David Bau, and Jun-Yan Zhu. Rewriting geometric rules of a gan. *ACM Transactions on Graphics (TOG)*, 41 (4):1–16, 2022.

Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8798–8807, 2018.

Lilian Weng. What are diffusion models? *lilianweng.github.io*, Jul 2021. URL https://lilianweng.github.io/posts/2021-07-11-diffusion-models/.

Mert Yuksekgonul, Maggie Wang, and James Zou. Post-hoc concept bottleneck models. *arXiv preprint arXiv:2205.15480*, 2022.

Rui Zhang, Xingbo Du, Junchi Yan, and Shihua Zhang. Decoupling concept bottleneck model. 2023.

Gen Model	FID
GAN	28.2
CGAN	14.4
ACGAN	14.9
InfoGAN	15.4
CB-GAN	23.8
Diffusion	15.9
CF-Diffusion	16.7
CB-Diffusion	15.9
VAE	17.1
CVAE	17.0
CB-VAE	17.5

Table 2: FID scores for different classes of generative models on Celeb-A.

Appendix

A. Concept Bottleneck Layer

A.1. CB layer with non-binary concepts

Continous Variables. Our discussion in the main draft mostly addressed binary variables. Here is discuss a simple extension of the current framework that also extends to continuous variables as well. First, we represent each continuous variable with a single context vector instead of the positive and negative context vectors. Second, we transform each continuous variable so that it is normalized to [0, 1]. The output of the probability vector is then the normalized continuous variable. Instead of the binary cross-entropy loss function as before, we use the mean-squared error loss function. With these changes, we can now account for continuous variables.

Joint Categorical Variables. This can be extended to categorical concepts by having j context vector where j is the number of categories, i.e., each concept is represented as $w_i^1, \ldots, w_i^j$; Ψ_i now predicts a probability for each class via a Softmax function $[\hat{c}_i^1 \ldots \hat{c}_i^j] = \Psi_i([w_i^1, \ldots, w_i^j]^T)$ the final context vector is constructed as the weighted mixture of all classes. In practice, we found that forcing sparse concept probabilities by adding temperature to the Softmax or using Gumbel-Softmax (Jang et al., 2016) improves performance since this makes the probabilities more aligned with the ground truth concept distribution. For continuous variables, the context network Ψ would generate a single context vector, which would then be concatenated directly into the final context embedding w; for interventions, an additional encoder is needed to encode the continuous variable to its context vector as done by (Shoshan et al., 2021).

A.2. Adapting CB layer to generative models

In Figure 4, we show where to add the CB layer for each type of generative model. For the VAE Figure 4-a, the CB layer is the final layer of the encoder, which means the output latent vector of the encoder is directly associated with pre-defined concepts. Similarly, in GANs, the input to the generator contains the pre-defined concepts (see Figure 4-b). Diffusion models typically follow a U-Net structure, and we insert the CB layer right after the middle block of the U-Net as shown in Figure 4-c.

B. Experiments

B.1. Setup

B.1.1. Datasets.

Color-MNIST We use a variation of MNIST introduced by Arjovsky et al. (2019). Where every digit has is either red or green. Here we assume we have 11 attributes (10 labels and 2 colors); each attribute has an active and inactive state. One can think of this as a multitasking problem where different tasks can exist in the same image.

Celeb-A (Liu et al., 2015): Following (Espinosa Zarlenga et al., 2022), we select the 8 most balanced attributes out of each image's 40 binary attributes, as defined by how close their distributions are to a random uniform binary distribution, and use attributes. Each image has 8 attributes each represented as active or inactive giving a total of 16 classes. We downsample every image to have shape (3, 64, 64).

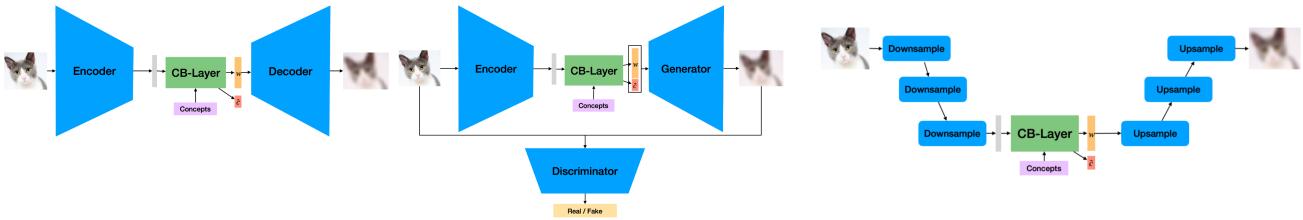
B.1.2. Baselines.

We consider three types of generative models: VAE, GANs, and diffusion models. We compare our method with each family's most commonly used framework fro conditional generation. For VAEs, we benchmark against conditional VAEs (CVAE) (Sohn et al., 2015). For GANs, we consider CGAN (Mirza and Osindero, 2014a), ACGAN (Odena et al., 2017), and InfoGAN (Chen et al., 2016). For diffusion models, we compare with classifier-free (CF) diffusion models (Ho and Salimans, 2022). Across each generative model family, we keep the underlying model architecture fixed.

B.2. Generation Quality

In this section, we demonstrate that the proposed concept bottleneck generative model does not incur a degradation in output quality compared to alternatives without constrained representations.

Experimental Setup. To test that a CB generative model does not



(a) VAE with CB layer (b) GAN with CB layer (c) Diffusion with CB layer

Figure 4: Adapting concept bottleneck layer to different generative model families.

affect generation quality, we compare the output of a CB generative model to standard models whose layers have not been constrained on Celeb-A (64×64 version). We keep the image resolution and data pre-processing steps fixed across all models and the underlying model architecture fixed for each generative model family. From each method, we generate 40K random samples; we calculate Frechet inception distance (FID[1]) (Heusel et al., 2017; Seitzer, 2020) between the synthetic images and the training dataset.

Results. Table 2 shows FID scores across generative model classes. We observe that the concept bottleneck versions of the generative models have FID scores that match those of standard alternatives. We note that the goal here is not to achieve state-of-the-art FID scores but to show that adding a CB layer does not harm generation.

B.3. Steering CB Generative Models

B.3.1. Multi-concept steerability

Here, we are interested in assessing whether various control strategies can simultaneously induce a change in multiple concepts. We start with only one concept, 'on' at $k = 1$, and gradually increase the number of 'on' concepts. Note that at each value of k, we generate all possible concept combinations for that k. For empirical evaluation of concept steerability, we calculate the accuracy of a concept classifier when that concept is turned 'on'. If multiple concepts are on, we calculate the average accuracy over all 'on' concept classifiers.

Results. Figure 5 shows images generated by CB-GAN, we can add and remove multiple concepts from the generated images using the CB layer. We report the steerability metric across five attributes in Table 3. For each setting of K, we intervene on the concept vector for one thousand inputs during generation and measure the induced change in attributes with attribute-independent classifiers. We find that CB-GAN has average concept accuracy around 30%, across all experiments, which indicates that CB-GAN steerability performance does not deteriorate with the addition of concepts. CB-GAN all outperforms the baselines for up till $K = 3$. In the diffusion model and VAEs setting, adding a CB layer outperforms the baseline across all values of K.

B.4. Interpretability

B.4.1. Interpreting the generated output

One of the most important open-ended questions in AI research is answering *'why did a black-box DNN make certain predictions?'*.

[1]We use the following code https://github.com/mseitzer/pytorch-fid to calculate the FID scores

Class	K=1	K=2	K=3	K=4	K=5
CGAN	7.6 ± 0.1	11.3 ± 0.4	18.1 ± 0.6	26.9 ± 0.8	35.7 ± 1.1
ACGAN	7.5 ±0.2	13.1 ± 0.5	25.0 ± 0.7	40.6 ± 0.7	55.0 ±0.6
infogan	6.8 ± 0.4	9.6 ± 0.4	16.2 ± 0.5	24.5 ± 0.6	32.6± 0.7
CB-GAN	29.3 ± 0.5	30.0 ± 0.4	31.0 ± 0.3	32.9 ± 0.4	36.0 ± 0.4
CF-Diffusion	7.58±2.6	13.1±0.9	17.6±2.6	21.3±1.9	29.6±3.4
CB Diffusion	14.32±1.9	19.3±3.5	27.6±1.4	36.2±3.3	39.8±2.7
CVAE	6.2±1.37	14.5±3.1	20.3±1.4	24.7±1.9	34.8±3.6
CB-VAE	15.1±2.46	18.9±2.3	27.6±4.8	38.3±3.7	41.6±0.9

Table 3: Multi-concept steerability metric.

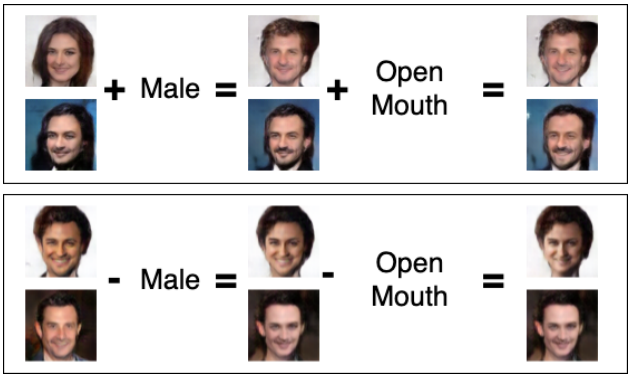


Figure 5: Adding and removing multiple concepts in CBGMs

One way to explain a black-box network is by using concept bottleneck layers (Koh et al., 2020; Espinosa Zarlenga et al., 2022). This generalizes to generative models as well. After adding a CB layer to the generative model one can understand why a model generated a particular output by looking at the concept probabilities. Figure 6 shows concept probabilities for different samples generated from CB-GAN on Celeb-A and color MNIST. Figure 7 shows concept probabilities for different samples generated from CB-VAE and CB-Diffuision on Celeb-A and color MNIST. By looking at each bar chart, one can understand which concepts were most effective in generating the output.

B.4.2. Model Debugging

In the main paper, we showed that we can use a validation dataset to check if the CB-layer is actually learning concepts during training. This can also be done by examining the concept training loss define in main paper Section 3.2. Figure 8-a shows the training loss for different models in the debugging experiment described in the main paper Section 4.4.2. Figure 8-a shows that both models were able to learn the green concept, i.e., training loss

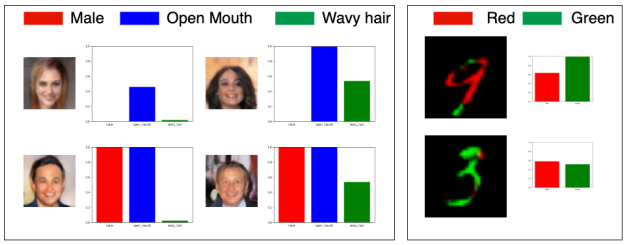


Figure 6: Sample-wise interpretability offered by adding a CB layer. By looking at the concept probability vector, we can understand why the model generated a particular output.

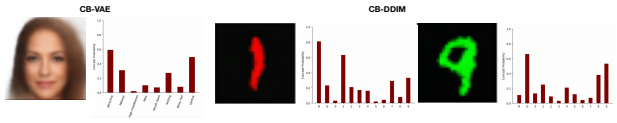


Figure 7: Sample-wise interpretability offered by adding a CB layer for Diffusion model and VAE. The concept probability vector helps us understand why the model generated a particular output.

decreases for both models. Figure 8-b shows that only Model-a's training loss decreased for the red concept, while the training loss for Model-b remained high.

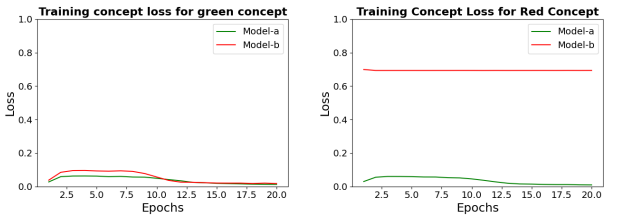


Figure 8: Model-a is trained on the ground truth concepts. Model-b is trained on the corrupted 'red' concept and the ground truth 'green' concept. (a) Low training loss on the green concept for both models. (b) Model-a has low training for the red concept, but Model-b's training loss does not decrease.

C. Related Work

In this work, we propose to insert a concept bottleneck (CB) layer into a generative model. Concept-based interpretability approaches have been applied post hoc (Kim et al., 2018; Yuksekgonul et al., 2022), but here we focus on incorporating using concepts during training. Concept bottleneck layers have been used for supervised learning (Koh et al., 2020; Chen et al., 2020) and semantic segmentation (Losch et al., 2019) but not generative models. Despite their benefits, CB layers are susceptible to feature leakage (Mahinpei et al., 2021; Margeloiu et al., 2021), which harms the ability to intervene on concepts. Recent work has proposed alternatives to address leakage and improve the performance of models based on CB layers (Havasi et al., 2022; Zhang et al., 2023). In this work, we incorporate these ideas into and adapt the concept embedding layer of Espinosa Zarlenga et al. (2022) into a generative model.

There is extensive literature on learning disentangled representations (Higgins et al., 2018); however, our goal is not to disentangle the generative model's latent space but to map the model's representations into human interpretable features. In general, there is often no guarantee that a disentangled representation should be human understandable.

A key capability that the CB layer allows is the ability to make model 'edits'. Model editing has taken on a renewed significance, with recent work demonstrating intriguing edits and control of Classifiers (Santurkar et al., 2021), GANs (Bau et al., 2020; Wang et al., 2021; 2022), and large language models (Raunak and Menezes, 2022; Mitchell et al., 2021; Meng et al., 2022). The current approach for model editing first searches the model's latent space to localize human-interpretable features before manipulating either the weights or activations that correspond to those features. However, a longstanding observation with large models is that they learn distributed representations, which makes effective localization challenging (Hase et al., 2023). Even if the localization strategy is effective, the latent representations of the generative model might not encode the interpretable feature that we seek to control. We circumvent these challenges by directly mapping to interpretable features, which obviates the need for a search.

D. Limitations

Our proposed CB generative model does come with certain challenges: it requires that the entire training set be annotated with pre-defined concepts that—a potentially laborious requirement in practice. Even though the CB layer can be applied broadly, we have only tested it for image tasks. Moving to text poses further challenges about what the nature of the concepts should be.