QUANTIFYING THE ACCURACY–INTERPRETABILITY TRADE-OFF IN CONCEPT-BASED SIDECHANNEL MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Concept bottleneck models make predictions through human-understandable concepts, providing interpretability. However, this bottleneck limits task accuracy when the concept set is incomplete or the concept-to-task map is insufficiently expressive. Recent concept-based models address this limitation by adding sidechannels that transmit additional information around the bottleneck. These sidechannels often recover accuracy, but they also raise a basic diagnostic question: how much can the final prediction still be interpreted using the concept pathway, and how much interpretability is lost? We study this question through a unified probabilistic view of concept sidechannel models. The framework exposes two inference modes: the default inference mode of the model, and a bottleneck mode, in which the sidechannel is replaced by an input-independent distribution. Comparing these modes yields an architecture-agnostic invariance metric, and naturally leads to a regularizer that penalizes discrepancies between the two modes. Across several architectures and datasets, we find that accuracy-only training often produces models whose predictions cannot be reproduced without the sidechannel, even when concept-only prediction is feasible. SIS regularization allows users to trace a controlled accuracy-interpretability frontier, improves responsiveness to concept interventions, and can yield task predictors that rely more visibly on concepts. Our results provide practical tools for auditing and tuning the role of sidechannels in concept-based models.

1 INTRODUCTION

Concept-based models (CBMs) have emerged as a promising direction for interpretable deep learning (Poeta et al., 2023; Koh et al., 2020; Espinosa Zarlenga et al., 2022; Mahinpei et al., 2021). CBMs achieve interpretability by incorporating high-level, human-understandable *concepts* explicitly within their architecture. A well-known example CBM is the Concept Bottleneck Model (Koh et al., 2020) (CBNM), which first predicts concepts (e.g. *whiskers*, *tail*) using a neural network, and then maps these predicted concepts to the target task using a linear layer (e.g. *cat*) (Figure 1, middle). This architecture makes predictions inherently explainable: one can trace decisions back to predicted concepts (e.g. 'the model predicted *cat* because it sees *whiskers* and *a tail*'), and the linear layer provides additional transparency by revealing the influence of each concept on the final prediction.

Despite their appeal, early CBMs such as CBNMs suffered from a significant drop in task accuracy compared to black-box models. This is because concepts in CBNMs form an *information bottleneck*: the model must rely solely on concepts to perform the task. In practice, it is often infeasible to design a concept set that fully captures all the information needed for accurate predictions, which makes a performance gap inevitable.

To address this limitation, recent works have augmented CBMs with an additional *sidechannel* that transmits extra information beyond concepts (Mahinpei et al., 2021; Espinosa Zarlenga et al., 2022; Barbiero et al., 2023; Sawada & Nakamura, 2022) (Figure 1).[1] This sidechannel can take different forms, such as embeddings (Espinosa Zarlenga et al., 2022) or one-hot masks (Debot et al., 2024),

[1] In the literature, this sidechannel is also often referred to as a "residual" or "sidepath".

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

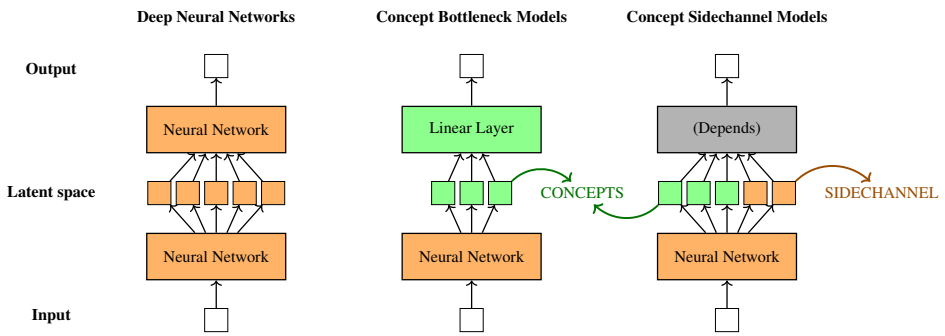


Figure 1: Comparison between Deep Neural Networks, (linear) Concept Bottleneck Models, and Concept Sidechannel Models. Interpretable components are **green**; uninterpretable components are **orange**. Concepts are interpretable because they are aligned to some human interpretation through direct supervision (e.g. *tail*, *brown hair*, *round*).

depending on the CBM in question. By leveraging this sidechannel, CBMs can close the accuracy gap with black-box models, often achieving similar performance irrespective of the used concept set.

However, this gain in accuracy comes at the cost of some interpretability. Unlike in CBNMs, predictions in sidechannel CBMs are no longer solely determined by concepts, but also by uninterpretable units of information (e.g. an embedding). The more a model relies on information captured by its sidechannel, the more accurate but less interpretable it becomes. In the extreme, the task predictor could rely completely on the sidechannel, bypassing the concepts entirely.

While some works have acknowledged this tension in a limited way by designing techniques that try to maximize concept usage (Kalampalikis et al., 2025; Havasi et al., 2022; Shang et al., 2024; Zabounidis et al., 2023), we show that this is a weaker criterion than minimizing sidechannel reliance, as a model could fully maximize its reliance on concepts while still relying heavily on the sidechannel to form the final prediction. The interpretability cost of using a sidechannel for CBMs in general is never quantified or explicitly optimized, and most evaluations and training objectives still focus on accuracy, leaving two key gaps in the literature: the lack of a *metric* to test for interpretability cost in sidechannel CBMs, and the lack of a principled way to *optimize* for this cost during training.

In this work, we aim to close these gaps. Our contributions are the following:

1. **Unified view:** We show that existing sidechannel CBMs can all be interpreted as different parametrizations of a single *meta-model*, which we formalize as a high-level probabilistic graphical model (PGM).
2. **Interpretability metric:** Using this PGM, we provide a natural evaluation method for sidechannel CBMs by equipping them with a *bottleneck mode*, where predictions only depend on the concepts. Based on bottleneck mode, we subsequently introduce the *Sidechannel Invariance Score* (SIS), which quantifies how often bottleneck mode reproduces the default predictions.
3. **Training objective:** We propose *SIS regularization*, a method that provides control over sidechannel CBMs' accuracy-interpretability trade-off by penalizing disagreement between the two modes.
4. **Interpretability discussion:** We discuss how different notions of interpretability in CSMs arise when considering both the expressivity of the predictor and its sidechannel reliance.

Through contributions (1), (2) and (4), we provide a clearer understanding of the structure and trade-offs of sidechannel CBMs. Through (3), we introduce practical ways for developing models that are not only accurate but also interpretable.

As sidechannel CBMs become more widely used, interpretability claims increasingly depend on an implicit assumption: that the sidechannel improves accuracy without dominating the decision. Our goal is to make this assumption testable.

2 BACKGROUND

Concept Bottleneck Models. Concept Bottleneck Models (CBNMs) (Koh et al., 2020) are CBMs that consist of two functions: a *concept predictor* ($X \to C$) that maps some low-level input features X (e.g. an image) to high-level, human-understandable concepts C (e.g. *whiskers*, *tail*), and a *task predictor* ($C \to Y$) that maps the concepts to some target task Y (e.g. *cat*). CBNMs are trained by supervising both concepts and task with the goal of aligning each concept to a human interpretation and obtaining high task performance. This supervision comes either from concept and task labels in the dataset or from vision-language models, the latter removing the need for human annotations (Oikarinen et al., 2023). Typically, CBNMs use a neural network as concept predictor and a linear layer or neural network as task predictor. A key downside of CBNMs is that their accuracy for predicting the task Y is limited by the employed concepts C, as they form an information bottleneck.

Concept Sidechannel Models. Concept Sidechannel Models (CSMs) are CBMs that address the information bottleneck issue by predicting Y not only using C but also using some additional information (Espinosa Zarlenga et al., 2022; Sawada & Nakamura, 2022; Yuksekgonul et al., 2022; Barbiero et al., 2023). This additional information comes in different forms for different CSMs. Some examples of this are an embedding predicted from X (Mahinpei et al., 2021) and a one-hot mask predicted from X (Debot et al., 2024). The central idea behind CSMs is primarily to achieve (near) black-box accuracy, and secondarily to be as interpretable as possible.

Notation. We write distributions of random variables in upper case (e.g. $p(X)$) and the variable assignments in lower case (e.g. $p(X = x)$). When it is clear from the context, we will abbreviate assignments (e.g. $p(x)$ means $p(X = x)$). Conditional distributions are written using a vertical bar (e.g. $p(Y|X)$ or $p(Y = y|X = x)$).

3 METHOD

In this section, we present our proposed method. We begin by clarifying the distinction between representation interpretability (interpretability of the intermediate representations) and functional interpretability (interpretability of the prediction process) (Section 3.1). Subsequently, we introduce a probabilistic *CSM meta-model*, a unified framework for representing all CSMs we identified (Section 3.2). We then show how this meta-model enables two modes of inference: the *default* mode and the *bottleneck* mode, in which the sidechannel is deactivated (Section 3.3). The delineation of these two modes naturally motivates the introduction of a novel metric, the Sidechannel Invariance Score (SIS), which quantifies the distance between the modes (Section 3.4). We then show how this distance can be employed as a regularization criterion for CSMs, providing explicit control over the accuracy-interpretability trade-off in CSMs. Finally, in Section 3.5, we discuss how the sidechannel not only supplements task-relevant information missing in the concepts, but also enhances concept-to-task expressivity for some CSMs, highlighting a trade-off between representation interpretability and functional interpretability.

3.1 REPRESENTATION INTERPRETABILITY VS FUNCTIONAL INTERPRETABILITY

Based on existing informal notions within the CBM community (Barbiero et al., 2025), we make two distinctions in interpretability for CBMs: *representation interpretability* and *functional interpretability*. We first consider the classical form of interpretability used in machine learning.

Definition 3.1 (Functional Interpretability of CBMs)**.** A CBM is *functionally interpretable* if and only if the mapping from the CBM's concepts (and sidechannel) to the task is an interpretable function.

What one considers an interpretable function is subjective to the human user. According to Rudin et al. (2022), standard cases of interpretable functions include linear layers, logic rules and small decision trees. For CSMs, only considering this form of interpretability is insufficient as CSMs also make their task prediction using uninterpretable representations (i.e. the sidechannel). Thus, we must also consider a form of interpretability that depends on the use of such uninterpretable representations.

Definition 3.2 (Representation Interpretability of CBMs)**.** A prediction is said to be *representation interpretable* if and only if it is derived exclusively from units of information that are themselves interpretable. A concept-based model is more *representation interpretable* if a larger fraction of its task-level predictions are representation-interpretable.

Here, "prediction" denotes the predicted class, not the predicted probabilities. "Derived exclusively" means that the predicted class is unchanged when uninterpretable units of information are removed. This should not be treated as functional or causal independence. Throughout the paper, "depends on," "relies on," or "uses" the sidechannel refers to this decision-level dependence.

We make the following assumption that is standard for CBMs regarding interpretable units:

Assumption 3.1 (Interpretable Units of Information in CBMs)**.** In a CBM, the concepts C are the only interpretable units, since they are explicitly supervised to align with some human understanding.[2] All other internal variables are uninterpretable, irrespective of their form, as they lack explicit human alignment (e.g. sidechannels).

Proposition 3.1 (Representation Interpretability of CBNMs and CSMs)**.** Under Assumption 3.1:

1. A CBNM is fully representation-interpretable irrespective of the used task predictor (e.g. linear layer or neural network): every prediction $\hat{y}$ depends only on the concepts.
2. A CSM is partially representation-interpretable: a prediction $\hat{y}$ is representation-interpretable iff removing the input-related information in the sidechannel does not affect $\hat{y}$.

Example 3.1. Consider a CSM with a single sidechannel neuron and a linear layer mapping the concepts and neuron to the task. If the class label y is directly encoded in the sidechannel neuron, the model can predict y by relying solely on this neuron. This model is completely not representation interpretable (since predictions fully rely on the sidechannel) but functionally interpretable (since the linear layer clearly shows how y is obtained). Conversely, if the sidechannel neuron carries no information (a dummy value), but the model uses a neural network on the concepts and neuron, then predictions are representation interpretable (since they depend only on concepts) but not functionally interpretable (due to the black-box predictor).

Current CSMs focus on achieving functional interpretability while maintaining high accuracy, and do not consider representation interpretability, despite its importance as illustrated above, as it is a necessary condition for a CSM being interpretable. **Therefore, our focus lies on testing for and improving representation interpretability in CSMs.**

3.2 The Meta-Model of CSMs

Many CSMs have been developed recently. While they differ in many ways, their high-level structure is very similar, and they can be understood as different instantiations of a common high-level model. We call this the *CSM meta-model*, which consists of 3 functions: a concept predictor $\phi_c : X \to C$, a sidechannel predictor $\phi_z : X \to Z$, and a task predictor $\phi_y : C, Z \to Y$. Both the functions (ϕ_c, ϕ_z, ϕ_y) and the variables (C, Z, Y) can differ across CSMs. For example, concepts may be continuous ($C = \mathbb{R}^{n_C}$) or discrete ($C = \{0, 1\}^{n_C}$). Sidechannels also vary in form, e.g. embeddings ($Z = \mathbb{R}^{|Z|}$) (Mahinpei et al., 2021) and one-hot masks ($Z = \{0, 1\}^{|Z|}$) (Debot et al., 2024).

Without loss of generality, the meta-model can be expressed as a probabilistic graphical model (PGM) (Figure 2). The PGM factorizes the joint distribution as:

$$p(y, c, z, x) = p(x) \cdot p(c|x) \cdot p(z|x) \cdot p(y|c, z), \tag{1}$$

with general task inference defined as

$$p(y|x) = \sum_{c,z} p(c|x) \cdot p(z|x) \cdot p(y|c, z) \tag{2}$$

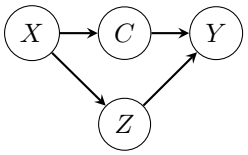


Figure 2: CSM Meta-Model.

Here, the summation is taken over all possible assignments of the concepts c and the sidechannel z. For continuous variables, the summations are replaced with integrals.

Specific CSMs then correspond to particular parametrizations of this PGM. Even models without probabilistic semantics can be cast into this framework (such as Concept Embedding Models (Espinosa Zarlenga et al., 2022)).[3] Two examples that we will work out in detail are CRM and CMR; for more examples, we refer to Appendix A.

[2] This assumes appropriate measures have been taken to prevent concept leakage (Marconato et al., 2022), which may otherwise misalign the concepts.

[3] Concepts and sidechannels can be represented as delta distributions. The summation in Equation 2 then reduces to a single evaluation due to the Dirac delta's sifting property.

Example 3.2 (Concept Residual Model (CRM) (Mahinpei et al., 2021))**.** Concepts are modelled as delta distributions: $p(c|x) = \delta(\hat{c} - c)$ where $\hat{c} = g(x)$ with g a neural network. The sidechannel z is an embedding represented by a delta distribution, i.e. $p(z|x) = \delta(\hat{z} - z)$ where $\hat{z} = f(x)$ with f a neural network. The task predictor $p(y|c, z)$ is a neural network. Due to the sifting property of the Dirac delta, the inference expression simplifies to a single evaluation of the task predictor: $p(y|x) = p(y|\hat{c}, \hat{z})$. CRM is not functionally interpretable due to the neural task predictor.

Example 3.3 (Concept Memory Reasoner (CMR) (Debot et al., 2024))**.** Concepts are modelled as Bernoulli random variables: $p(c|x) = g(x)$ with g a neural network with sigmoid activation. The sidechannel z is a categorical distribution representing a one-hot mask: $p(z|x) = f(x)$ with f a neural network with softmax activation. The task predictor $p(y|c, z)$ possesses a set of learned logic rules. At inference time, it evaluates only the rule indicated by the sidechannel z on the concepts. The resulting inference expression is: $p(y|x) = \sum_{c\in\{0,1\}^{n_C}} \sum_{z=1}^{n_R} p(c|x) \cdot p(z|x) \cdot p(y|c, z)$, where n_R is the number of learned rules. CMR is functionally interpretable due to its rule-based task predictor.

3.3 Default and Bottleneck Modes for CSMs

In this section, we operationalize Definition 3.2 for CSMs. Consider a CSM where the sidechannel $p(z|x)$ is replaced with an input-independent distribution $p(z)$. The task prediction then depends on x only through the concepts, i.e. $p(y|c, x) = p(y|c)$. This disables the transmission of input-specific information through the sidechannel, ensuring that predictions are fully representation-interpretable.

Thus, we can define two CSM operation modes, **Default Mode** and **Bottleneck Mode**:

$$\textbf{Default Mode:} \quad p_{\theta,\phi,\psi}(y|x) = \sum_{c,z} p_\theta(c|x) \cdot \boxed{p_\phi(z|x)} \cdot p_\psi(y|c, z) \tag{3}$$

$$\textbf{Bottleneck Mode:} \quad p_{\theta,\gamma,\psi}(\bar{y}|x) = \sum_{c,z} p_\theta(c|x) \cdot \boxed{p_\gamma(z)} \cdot p_\psi(y|c, z) \tag{4}$$

where the subscripts $\{\theta, \phi, \psi, \gamma\}$ indicate learnable parameters. Notice that the two modes differ *exclusively* on whether their predictions use $p_\phi(z|x)$ or $p_\gamma(z)$. We can always make predictions using either mode, with bottleneck mode guaranteeing full representation interpretability. Note that default mode corresponds to the standard inference of CSMs (Equation 2), and that bottleneck mode corresponds to the do-intervention $p(\bar{y}|x) = p(y|x, do(z \sim p(z)))$.

A natural question is how to obtain $p(z)$. A sensible choice is to marginalize over the input distribution and approximate this using the training dataset $\mathcal{D}$, since the true data distribution $p(X)$ is typically unknown: $p_\gamma(z) = \sum_x p(x) \cdot p_\phi(z|x) \approx \frac{1}{|\mathcal{D}|} \sum_{x\in\mathcal{D}} p_\phi(z|x)$. While this choice for $p(z)$ minimizes the expected KL divergence between $p(z)$ and $p(z|x)$ (see Theorem E.5 in Appendix E.4), it only minimizes an upper bound on the difference between default mode and bottleneck mode. Therefore, a better approach is to *learn* a prior $p(z)$ such that Bottleneck Mode mimics Default Mode as much as possible. We will explain how to learn this prior in Section 3.4.

3.4 Measuring and Optimizing for Representation Interpretability

The definition of these two modes allows us to devise a metric to measure how often a CSM's predictions survive the removal of input-specific sidechannel information by checking the agreement between predictions from the two modes.

Definition 3.3 (Sidechannel Invariance Score - SIS)**.** Let y_x denote the prediction from default mode and, for an input-independent distribution $q \in \Delta(\mathcal{Z})$, let $\bar{y}_x^q$ denote the prediction from bottleneck mode with $p(z) = q$, both obtained by taking the most likely class predicted by the corresponding distributions $p(y \mid x)$ and $p(\bar{y} \mid x)$. We define the *Sidechannel Invariance Score* (SIS) as

$$\text{SIS} = \sup_{q\in\Delta(\mathcal{Z})} \mathbb{E}_{x\sim p(X)}\big[\mathbb{1}[\, y_x = \bar{y}_x^q \,]\big].$$

SIS concerns the predicted class rather than equality of the predictive distributions, which makes it pragmatic for human users. It measures the frequency with which the model's prediction stays the same when input-specific sidechannel information is removed, i.e. how invariant it is to the sidechannel. The supremum is over input-independent distributions, so no $p(z)$ can transmit information

about x. As a consequence, *SIS exactly measures representation interpretability as formalized in Definition 3.2, while approximations of SIS provide a certificate (see Appendix B)*. If SIS is high, the CSM's predictions can often be interpreted through bottleneck mode. Note that it does not certify that the default computation was carried out through the concepts. Accordingly, when we describe a CSM as more or less representation interpretable in what follows, we mean as measured by SIS.

In practice, SIS cannot be computed exactly because (1) the supremum is intractable and (2) the true data distribution $p(X)$ is unknown. We approximate (1) by fitting a prior $p_\gamma(z)$ restricted to the model's own distributional form for Z (e.g. categorical, delta), optimizing it to approximate the optimal prior as closely as possible. We approximate (2) using a test set $\mathcal{D}$: $\widehat{\text{SIS}} = \frac{1}{|\mathcal{D}|}\sum_{x\in\mathcal{D}} \mathbb{1}[y_x = \bar{y}_x]$. We can then provide a formal lower bound on the model's SIS using Hoeffding's Inequality.

Theorem 3.1 (Generalization of SIS)**.** *Let $\widehat{SIS}$ be the empirical SIS computed of n i.i.d. samples, using a prior $p_\gamma(z)$. For every $\delta \in (0, 1)$, with probability at least $1 - \delta$ over the unseen test sample,*

$$\text{SIS} \geq \widehat{\text{SIS}} - \sqrt{\frac{\log(1/\delta)}{2n}}.$$

For the proof, see Appendix E.2.

Most CSMs are currently trained to maximize accuracy (see Section 4). Our meta-model enables explicit optimization for representation interpretability by introducing a loss term that penalizes discrepancies between predictions from default mode ($p(y|x)$) and bottleneck mode ($p(\bar{y}|x)$). Any suitable divergence, such as KL divergence, can be used. This approach integrates seamlessly into any CSM. For example, when maximizing the likelihood of training data, the objective becomes:

$$\text{argmax}_{\phi,\psi,\gamma,\theta}\Big[\sum_{(x,c,y)\in\mathcal{D}} (\log p_{\phi,\psi}(y|c,x) + \alpha \cdot \log p_\theta(c|x) - \beta \cdot \text{DIV}(p_{\phi,\psi}(y|c,x)||p_{\gamma,\psi}(\bar{y}|c,x)))\Big]$$

where α and β are hyperparameters, and $\text{DIV}(\cdot||\cdot)$ denotes a chosen divergence measure. We refer to this additional term as **SIS regularization**. It can also be incorporated into alternative training schemes, such as sequential or joint CBM training (Koh et al., 2020). By using SIS regularization, one can control the accuracy-interpretability trade-off of CSMs by tuning β. We provide a lower bound for SIS in terms of KL divergence in Theorem E.4 in Appendix E.3. We also discuss the alternative of directly regularizing Z in Appendix E.4, showing with Theorem E.6 that regularizing Z is a strictly (and unnecessarily) stronger criterion that may hurt performance for some CSMs. Finally, evaluating an already trained model with SIS can be done by optimizing only SIS regularization while keeping all parameters frozen except $p_\gamma(z)$. Note that the prior $p_\gamma(z)$ is learned on training data.

Importantly, this method is applicable to any CSM, regardless of its parametrization within our meta-model PGM (e.g. choice of sidechannel z or form of $p(y|c, z)$).

3.5 Expressivity of CSMs in Bottleneck Mode

In default mode, CSMs are typically universal classifiers: they are as expressive as neural networks for classification. This expressivity comes from the sidechannel, which can allow them to learn any mapping from input to task ($X \to Y$), irrespective of the employed concept set.

When in bottleneck mode, however, their ability to learn input-to-task mappings is limited by the quality of the concept set. Because concept sets differ between datasets, a general comparison between CSMs cannot be made. Instead, we can analyze their expressivity in learning mappings from concepts to task ($C \to Y$). Some CSMs are more expressive than others in this regard, for instance: CRM (Mahinpei et al., 2021) applies a neural network to the concepts. This is fully expressive but not functionally interpretable. CBM-AUC (Sawada & Nakamura, 2022) applies a linear layer to the concepts, which is functionally interpretable but not expressive. CMR (Debot et al., 2024) falls in between, as it applies a set of learned logic rules to the concepts. Its expressivity increases with the number of learned rules. For more examples, we refer to Appendix A.

Prior work primarily motivates sidechannels as compensation for insufficient concepts. Here, we highlight a distinct role. The difference in $C \to Y$ expressivity between default and bottleneck mode for some CSMs implies that they do not only use the sidechannel to use information related to Y that cannot be found in C, but also to improve their $C \to Y$ expressivity. This yields two important

insights. First, to match accuracy on expressive tasks Y, a functionally interpretable but inexpressive CSM (e.g. CBM-AUC) must rely more on its sidechannel than a non-functionally interpretable but expressive CSM (e.g. CRM). Thus, the former may achieve higher functional interpretability but lower representation interpretability than the latter. Second, even if the concepts are sufficient for the task (i.e. a perfect predictor exists given a sufficiently expressive model), an inexpressive functionally interpretable CSM still needs to use its sidechannel to achieve high accuracy. We formalize this (Appendix E, Theorem E.2) and show this empirically (Section 5).

Example 3.4. Consider two binary concepts and a task defined as their logical XOR ($y := c_1 \oplus c_2$). Suppose a CSM uses a linear layer to map the concepts and a sidechannel (some neurons) to y. In bottleneck mode, the CSM cannot predict y accurately, since it is not linearly separable. In default mode, the model can encode features like $c_1 \wedge \neg c_2$ and $\neg c_1 \wedge c_2$ into the sidechannel. The linear layer can then compute the logical OR of these features to solve the XOR task. The sidechannel effectively extends the concept bottleneck with concept combinations to increase the model's expressivity.

4 Related Work

While explicitly measuring and optimizing representation interpretability has not been studied directly, several related research directions pursue overlapping but distinct goals. Most of these works focus on encouraging models to use concepts extensively, whereas our focus lies in minimizing reliance on the sidechannel. Note that the latter entails the former, but goes even further.

A major line of research in CBMs emphasizes *intervenability* (Espinosa Zarlenga et al., 2022; 2023; Havasi et al., 2022): replacing concept predictions with their ground truth should improve downstream task accuracy. However, a model may rely heavily on both concepts and sidechannel. In this case, intervenability is high even though representation interpretability is low.

Some works *disentangle* the sidechannel from the concepts (Zabounidis et al., 2023). This can improve representation interpretability by removing concept information from the sidechannel, but does not reduce sidechannel usage beyond this. Moreover, it can substantially harm accuracy in CSMs that need sidechannels for expressivity (e.g. Debot et al. (2024)), despite not being necessary for achieving representation interpretability (see Appendix C).

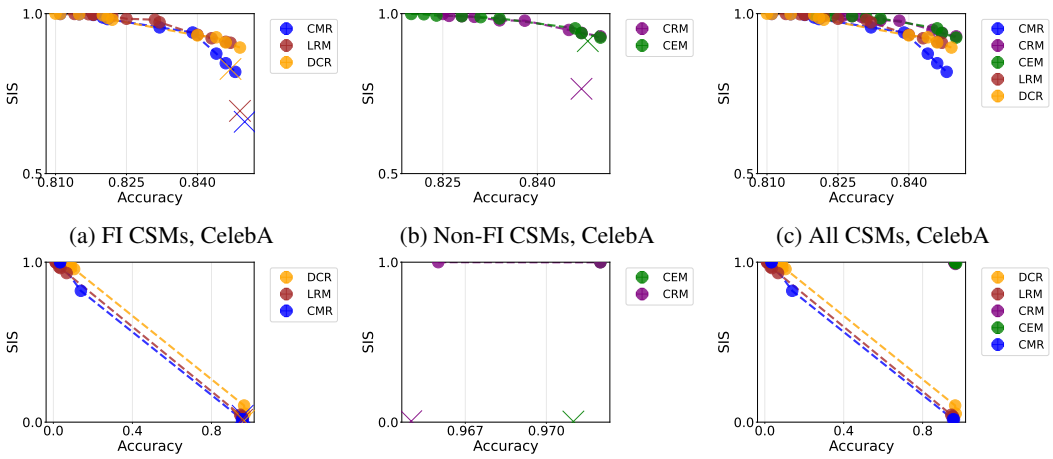
Other works encourage *concept usage* during prediction (Kalampalikis et al., 2025; Shang et al., 2024), which does not directly address sidechannel use (see Section 5), and may require structural assumptions of the model (e.g. a factorizable task predictor, (Shang et al., 2024)). Zhang et al. (2024) propose a regularizer that reduces sidechannel use, but only for their factorized task predictor, which our approach does not require. Finally, Appendix C.2 compares SIS with the concept completeness score (Havasi et al., 2022) and Zhang et al. (2024)'s interpretability metric.

5 Experiments

This section provides essential information about the experiments (see Appendix D for details). We report results averaged over three seeds, and we give standard-deviations as shaded areas. For Pareto curves, we only give the results for the first seed (see Appendix D for the remaining ones).

Datasets. We use three standard CBM datasets: CelebA (Liu et al., 2015), CUB (Welinder et al., 2010), and MNIST-Add (Manhaeve et al., 2018). CelebA's concepts are insufficient for the task; MNIST-Add's are sufficient but has an expressive task. CUB's are sufficient, has an inexpressive task, but has difficult concept prediction. For additional results on CUB, we refer to Appendix D.2.

Models. We use the following CSMs: Concept Residual Models (CRM) (Mahinpei et al., 2021), Concept Embedding Models (CEM) (Espinosa Zarlenga et al., 2022), Deep Concept Reasoner (DCR) (Barbiero et al., 2023), and Concept Memory Reasoner (CMR) (Debot et al., 2024). We define Linear Residual Model (LRM) as CRM but with a linear layer task predictor. In bottleneck mode, DCR, CMR, and LRM are *functionally interpretable* but *inexpressive*, while CRM and CEM are *not functionally interpretable* but *expressive*. As **baselines**, we adapt concept usage methods (Kalampalikis et al., 2025; Shang et al., 2024) to CRM (which we call CRM* and CRM**, respectively).



(a) FI CSMs, CelebA (b) Non-FI CSMs, CelebA (c) All CSMs, CelebA

(d) FI CSMs, MNIST-Add (e) Non-FI CSMs, MNIST-Add (f) All CSMs, MNIST-Add

Figure 3: Accuracy vs SIS trade-off in CSMs. Each point is a hyperparameter configuration, keeping only pareto-efficient points. Crosses are the most accurate configuration trained without SIS regularization (not in (c) and (f)). FI means "functionally interpretable". In (e), CEM with regularization overlaps with the right-most point of CRM.

Table 1: Hypervolumes of the Pareto curves, reported as mean ± standard deviation over three seeds. CRM* and CRM** are CRM-like concept usage approaches. CMR and DCR do not natively support CUB's multiclass classification.

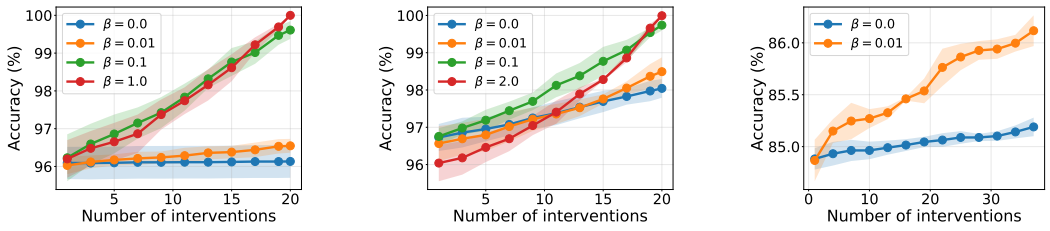
	CelebA		MNIST-Add		CUB	
Model	No SIS reg.	SIS reg.	No SIS reg.	SIS reg.	No SIS reg.	SIS reg.
CEM	20.4 ± 0.0	22.3 ± 0.1	0.4 ± 0.1	87.5 ± 0.2	8.4 ± 0.2	63.1 ± 0.5
CRM	18.0 ± 1.0	22.3 ± 0.1	0.6 ± 0.1	87.3 ± 0.4	0.4 ± 0.0	61.7 ± 0.1
CRM*	21.9 ± 0.0	–	86.5 ± 0.4	–	42.4 ± 0.4	–
CRM**	20.3 ± 0.2	–	86.6 ± 0.9	–	40.0 ± 0.1	–
LRM	16.1 ± 0.5	22.2 ± 0.2	1.5 ± 0.3	5.0 ± 1.7	0.5 ± 0.2	51.8 ± 0.4
CMR	15.6 ± 0.9	22.1 ± 0.0	3.2 ± 0.3	4.7 ± 0.5	–	–
DCR	18.9 ± 0.5	22.2 ± 0.0	1.1 ± 0.3	10.8 ± 1.0	–	–

5.1 Results and Discussion

We consider the following research questions: How do state-of-the-art CSMs score under SIS? How much does SIS regularization improve this? What is the trade-off w.r.t. accuracy? How do methods encouraging concept usage for CRM-like models (Section 4) fare regarding SIS? (Side-effects) Does SIS regularization improve intervenability or the quality of interpretable task predictors?

Current optimization of CSMs can result in uninterpretable models (Figure 3, crosses), contrary to when using SIS regularization (Figure 3, circles). Models that are optimized only for accuracy have a low SIS, meaning no input-independent prior allows to reproduce (and interpret) their predictions. Notably, this is even the case when the sidechannel is completely unnecessary, e.g. when training expressive CSMs on datasets with sufficient concepts (Figure 3e). Conversely, SIS regularization moves CSMs onto accuracy-SIS frontiers that dominate the unregularized models, letting the human choose a point on the frontier. Interestingly, unregularized CEM on CelebA already approaches the frontier. This observation also highlights the value of SIS: it allows us to establish empirically when existing CSMs are representation-interpretable in practice, which was not possible before. Table 1 gives aggregated results over all seeds.

When sidechannel usage is unnecessary, SIS regularization ensures CSMs whose predictions do not require it (Figure 3e). With sufficient concepts, expressive CSMs can reach high accuracy without predictions requiring the sidechannel, matching Concept Bottleneck Models on this criterion.



(a) CRM on MNIST-Addition (b) CEM on MNIST-Addition (c) LRM on CelebA

Figure 4: Intervenability with ($\beta > 0$) and without ($\beta = 0$) SIS reg. for different reg. weights β. The y-axis denotes task accuracy after intervening on a number of concepts denoted by the x-axis.

Inexpressive CSMs require the sidechannel for expressive tasks (Figure 3d). Despite MNIST-Add having sufficient concepts, the linear expressivity of these CSMs' bottleneck mode makes them either very accurate but completely not representation interpretable, or completely representation interpretable but very inaccurate. Additionally, and interestingly, our analysis shows that CMR seems unable to exploit its non-linear expressivity. In Appendix D, we address this by replacing its rule learner with an existing alternative, enabling near-perfect accuracy and interpretability on this task.

Adapted existing concept usage approaches yield similar or smaller gains in representation interpretability (Table 1). Using SIS, we are able measure how effective these existing techniques are in making the CSM more interpretable. While they also increase representation interpretability, their improvements are similar or smaller than those achieved with SIS regularization.

SIS regularization increases intervenability (Figure 4). As SIS regularization reduces sidechannel reliance in the CSM, an automatic advantage is that the CSM relies more on the concepts to predict the task, making it more responsive to concept interventions. For more curves, see Appendix D.

SIS regularization improves task predictor quality of functionally interpretable CSMs (Figures 5 and 12, Appendix D.2). We compare the most accurate LRMs trained with and without regularization on CelebA (84.8% vs 84.9% accuracy). Without SIS regularization, many of the most contributing weights of LRM's linear task predictor are uninterpretable latent factors from the sidechannel (e.g. *z11*), harming interpretability. In contrast, with SIS regularization, large weights correspond to meaningful concepts (e.g. *Gray Hair*) (Figure 5a), and the total weight magnitude shifts significantly toward concepts rather than the sidechannel (Figure 5b). We also quantitatively show the positive effect of SIS regularization on the linear task predictor's weights in Figure 12 in Appendix D.2.

6 Conclusion

We addressed a fundamental gap in CSMs: the lack of principled methods to measure and control the accuracy-interpretability trade-off. We proposed a meta-model that places existing CSMs within a single framework, enabling us to disconnect representation interpretability from functional interpretability. Building on this, we introduced SIS as a measure for any CSM's representation interpretability, and demonstrated using it as a regularization objective. Our experiments reveal that existing CSMs, when trained using typical objectives, often have low SIS, and that SIS regularization produces models that are more representation interpretable and responsive to interventions. Our analysis allowed us to find a weakness in a state-of-the-art CSM (CMR), which we addressed. Our contributions provide a unifying formalization and practical tools for developing interpretable CSMs.

Limitations. SIS is a decision-level invariance criterion: it certifies that a concept-only inference mode reproduces the default predictions, not that the default computation was carried out through the concepts. SIS does not replace metrics like accuracy and intervenability, and does not make claims regarding faithfulness to the original computational pathway (e.g. functional or causal dependencies). Because SIS' supremum is approximated by a learned prior, the population agreement achieved by this prior is a lower bound whose tightness depends on optimization, and its empirical estimate is subject to the finite-sample guarantee in Theorem 3.1. Furthermore, interpretability is a broad concept, and SIS only measures one form of it (representation interpretability). Finally, we build upon the standard assumption of CBMs that concepts are the only interpretable units of information.

AI USE STATEMENT

In this work, we used generative AI tools to aid or polish writing and for proving mathematical claims. We have not used generative AI tools for other tasks. We have reviewed all AI-assisted work. We manually checked every sentence that was co-written by LLMs and every proof. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide details for reproducibility in Appendix D, and will publish the code publicly upon acceptance. Proofs of theoretical results are provided in Appendix E.

REFERENCES

Pietro Barbiero, Gabriele Ciravegna, Francesco Giannini, Mateo Espinosa Zarlenga, Lucie Charlotte Magister, Alberto Tonda, Pietro Lio', Frederic Precioso, Mateja Jamnik, and Giuseppe Marra. Interpretable neural-symbolic concept reasoning. In *ICML*, 2023.

Pietro Barbiero, Giuseppe Marra, Gabriele Ciravegna, David Debot, Francesco De Santis, Michelangelo Diligenti, Mateo Espinosa Zarlenga, and Francesco Giannini. Neural interpretable reasoning. *arXiv preprint arXiv:2502.11639*, 2025.

David Debot, Pietro Barbiero, Francesco Giannini, Gabriele Ciravegna, Michelangelo Diligenti, and Giuseppe Marra. Interpretable concept-based memory reasoning. *Advances of neural information processing systems 37, NeurIPS 2024*, 2024.

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

Mateo Espinosa Zarlenga, Katie Collins, Krishnamurthy Dvijotham, Adrian Weller, Zohreh Shams, and Mateja Jamnik. Learning to receive help: Intervention-aware concept embedding models. *Advances in Neural Information Processing Systems*, 36:37849–37875, 2023.

Marton Havasi, Sonali Parbhoo, and Finale Doshi-Velez. Addressing leakage in concept bottleneck models. *Advances in Neural Information Processing Systems*, 35:23386–23397, 2022.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Nektarios Kalampalikis, Kavya Gupta, Georgi Vitanov, and Isabel Valera. Towards reasonable concept bottleneck models. *arXiv preprint arXiv:2506.05014*, 2025.

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*, pp. 5338–5348. PMLR, 2020.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *ICCV*, 2015.

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.

Robin Manhaeve, Sebastijan Dumancic, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. DeepProbLog: Neural Probabilistic Logic Programming. In *NeurIPS*, pp. 3753–3763, 2018.

Emanuele Marconato, Andrea Passerini, and Stefano Teso. Glancenets: Interpretable, leak-proof concept-based models. *Advances in Neural Information Processing Systems*, 35:21212–21227, 2022.

Tuomas Oikarinen, Subhro Das, Lam Nguyen, and Lily Weng. Label-free concept bottleneck models. In *International Conference on Learning Representations*, 2023.

Eleonora Poeta, Gabriele Ciravegna, Eliana Pastor, Tania Cerquitelli, and Elena Baralis. Concept-based explainable artificial intelligence: A survey. *arXiv preprint arXiv:2312.12936*, 2023.

Cynthia Rudin, Chaofan Chen, Zhi Chen, Haiyang Huang, Lesia Semenova, and Chudi Zhong. Interpretable machine learning: Fundamental principles and 10 grand challenges. *Statistic Surveys*, 16:1–85, 2022.

Yoshihide Sawada and Keigo Nakamura. Concept bottleneck model with additional unsupervised concepts. *IEEE Access*, 10:41758–41765, 2022.

Chenming Shang, Shiji Zhou, Hengyuan Zhang, Xinzhe Ni, Yujiu Yang, and Yuwang Wang. Incremental residual concept bottleneck models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11030–11040, 2024.

Peter Welinder, Steve Branson, Takeshi Mita, Catherine Wah, Florian Schroff, Serge Belongie, and Pietro Perona. Caltech-ucsd birds 200. Technical Report CNS-TR-201, Caltech, 2010. URL `/se3/wp-content/uploads/2014/09/WelinderEtal10_CUB-200.pdf,http://www.vision.caltech.edu/visipedia/CUB-200.html`.

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

Renos Zabounidis, Ini Oguntola, Konghao Zhao, Joseph Campbell, Simon Stepputtis, and Katia Sycara. Benchmarking and enhancing disentanglement in concept-residual models. *arXiv preprint arXiv:2312.00192*, 2023.

Rui Zhang, Xingbo Du, Junchi Yan, and Shihua Zhang. The decoupling concept bottleneck model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

Supplementary Material

TABLE OF CONTENTS

A Overview of state-of-the-art CSMs **13**

B Learnable prior $p(z)$ **14**

C Limits of Disentanglement and Concept Completeness Score **14**

C.1 Disentangling for representation interpretability 14

C.2 Concept Completeness Score vs Sidechannel Invariance Score 15

D Experiments **16**

D.1 Experimental details 16

D.2 Additional results 19

E Additional theoretical results **21**

E.1 Unavoidable Reliance from insufficiency and inexpressivity 23

E.2 Proof of Theorem 3.1 24

E.3 Link between SIS regularization and SIS 24

E.4 The sidechannel prior and information contraction 25

F Code and licenses **26**

A OVERVIEW OF STATE-OF-THE-ART CSMS

In this section, we give more examples of how state-of-the-art CSMs are parametrizations of our meta-model.

We do not give extensive details about the models, only a high-level overview. We also take the examples CRM and CMR again as they appear in the main text.

Concept Residual Model (CRM) (Mahinpei et al., 2021). In CRM, concepts are represented as delta distributions: $p(c|x) = \delta(\hat{c} - c)$ where $\hat{c} = g(x)$ with g a neural network. The sidechannel z is an embedding also represented by a delta distribution, i.e. $p(z|x) = \delta(\hat{z} - z)$ where $\hat{z} = f(x)$ with f a neural network. The task predictor $p(y|c, z)$ is a neural network. Due to the sifting property of the Dirac delta, the inference expression simplifies to a single evaluation of the task predictor: $p(y|x) = p(y|\hat{c}, \hat{z})$. CRM is not functionally interpretable due to the neural task predictor.

Concept Embedding Models (CEM) (Espinosa Zarlenga et al., 2022). In CEM, concepts are represented as delta distributions: $p(c|x) = \delta(\hat{c} - c)$ where $\hat{c} = g(x)$ with g a neural network. The sidechannel z consists of 2 embeddings per concept, each represented by a delta distribution, i.e. for $i \in \{1..n_C\}$ and $j \in \{1, 2\}$, $p(z_{i,j}|x) = \delta(\hat{z}_{i,j} - z_{i,j})$ where $\hat{z}_{i,j} = f_{ij}(x)$ with f_{ij} a neural network. The task predictor $p(y|c, z)$ first mixes the two embeddings of each concept using the concept scores ($z_i = c_i \cdot z_{i,1} + (1 - c_i) \cdot z_{i,2}$), and the resulting embeddings are concatenated and fed through a neural network. Due to the sifting property of the Dirac delta, the inference expression simplifies to a single evaluation of the task predictor: $p(y|x) = p(y|\hat{c}, \hat{z})$. CEM is not functionally interpretable due to the neural task predictor.

Concept Memory Reasoner (CMR) (Debot et al., 2024). In CMR, concepts are modelled as Bernoulli random variables: $p(c|x) = g(x)$ with g a neural network with sigmoid activation. The sidechannel z is a categorical distribution representing a one-hot mask: $p(z|x) = f(x)$ with f a neural network with softmax activation. The task predictor $p(y|c, z)$ encodes a set of learned logic rules. At inference time, it evaluates only the rule indicated by the sidechannel z on the concepts. The resulting inference expression is: $p(y|x) = \sum_{c \in \{0,1\}^{n_C}} \sum_{z=1}^{n_R} p(c|x) \cdot p(z|x) \cdot p(y|c, z)$, where n_R is a hyperparameter signalling CMR's number of learned rules. CMR is functionally interpretable due to the rule-based task predictor.

Deep Concept Reasoner (DCR) (Barbiero et al., 2023). In DCR, concepts are modelled as delta distributions: $p(c|x) = \delta(\hat{c} - c)$ where $\hat{c} = g(x)$ with g a neural network. The sidechannel z represents a fuzzy logic rule, given by 2 fuzzy values per concept, signalling the relevance and polarity of each concept in the rule: for $i \in \{1..n_C\}$ and $j \in \{1, 2\}$, $p(z_{i,j}|x) = \delta(\hat{z}_{i,j} - z_{i,j})$ where $\hat{z}_{i,j} = f_{ij}(x)$ with f_{ij} a neural network with sigmoid activation. The task predictor $p(y|c, z)$ uses fuzzy logical inference to deduce the task from the fuzzy logic rule and the predicted concepts. DCR is functionally interpretable due to the rule-based task predictor.

Concept Bottleneck Model with Additional Unsupervised Concepts (CBM-AUC) (Sawada & Nakamura, 2022). In CBM-AUC, concepts are also delta distributions. The sidechannel consists of two components: some unsupervised concepts and a weight for each concept and unsupervised concept. Both are represented by delta distributions. The task predictor computes the dot product of all concepts and weights. CBM-AUC is functionally interpretable due to the linear task predictor.

Hybrid Post-hoc Concept Bottleneck Models (PCBM-h) (Yuksekgonul et al., 2022). In PCBM-h, concepts are delta distributions, and the sidechannel is a single value denoted by a delta distribution. The task predictor applies some interpretable predictor (e.g. linear layer) to the concepts, providing a score y_1, which is summed with the value predicted by the sidechannel. PCBM-h is functionally interpretable due to the interpretable predictor followed by a simple sum.

Expressivity. Of these models, CRM and CEM are universal classifiers ($C \to Y$) in bottleneck mode, but not functionally interpretable. DCR and CBM-AUC are functionally interpretable, but are inexpressive ($C \to Y$) in bottleneck mode. PCBM-h is functionally interpretable but its expressivity depends on the used predictor. CMR is functionally interpretable and its expressivity in bottleneck mode is limited only by the number of rules it uses.

B LEARNABLE PRIOR $p(z)$

We propose to use a learnable prior for $p(z)$ as opposed to a marginalization approach. For some models, avoiding the marginalization can improve expressivity. For instance, for CMR, using the marginalization approach means that the final task prediction is still using a single logic rule in each complete assignment to all random variables. As a consequence, CMR's expressivity ($C \to Y$) is limited. By instead allowing CMR to use the entire set of learned rules (which would be impossible by the marginalization approach, as this value for z is out-of-distribution), CMR's expressivity becomes bounded only by the number of rules it has learned. Making z choose the entire rule set instead of a single rule can still be represented by the same categorical random variable z, but instead of selecting over rule sets that only contain single rules, it is allowed to pick rule sets containing multiple rules (which the original CMR cannot do, i.e. it is out-of-distribution in that sense). Moreover, using a marginalization approach would require a large computational effort for some CSMs. For instance, for CSMs that parametrize the sidechannel as a delta distribution, the empirical marginal would be a mixture of delta distributions with one component per dataset example. Evaluating bottleneck mode under this prior could require, for each inference query, a forward pass *per dataset example*.

On the learnable prior. One might worry that, because $p_\gamma(z)$ is learned, a high SIS can be obtained by fitting the prior rather than by predicting from concepts. It cannot: the prior is input-independent, so no γ can reintroduce information about x, and a prediction that genuinely requires input-specific sidechannel content is reproducible by no input-independent distribution. SIS is therefore defined as a supremum over $q \in \Delta(\mathcal{Z})$, and the population agreement obtained with any fitted $p_\gamma(z)$ is a *lower bound* on it. This remains true when we optimize within the model's own distributional form for Z (e.g. categorical, delta), a subset of $\Delta(\mathcal{Z})$. Better optimization can only expose representation interpretability that is already present, never fabricate it. Both ingredients belong to the claim: a model for which no such q exists is one whose predictions cannot be accounted for by its concepts, which is what SIS should report. We fit $p_\gamma(z)$ on training and validation data only, as Theorem 3.1 requires.

Out-of-distribution sidechannel values. The prior need not lie in the support of $\{p_\phi(z \mid x)\}_x$. Bottleneck mode requires input-independence, not on-manifoldness. This holds across architectures: for CRM and CEM the prior is an unconstrained vector in $\mathbb{R}^{|Z|}$, giving *more* freedom to leave the encoder's manifold than e.g. CMR's prior, which stays within the simplex over its own rules. It also does not change the model class: $p_\theta(c \mid x)$ and $p_\psi(y \mid c, z)$ are frozen, so the bottleneck classifier ranges over $\{\, c \mapsto \sum_z q(z)\, p_\psi(y \mid c, z) \,:\, q \in \Delta(\mathcal{Z}) \,\}$, a family fixed by the model's own task predictor, every member of which is a function of the concepts alone. Because SIS is a supremum over this family (by learning the best prior $p(z)$) rather than the effect of a perturbation of fixed magnitude, its value carries the same meaning, and is comparable, across architectures.

C LIMITS OF DISENTANGLEMENT AND CONCEPT COMPLETENESS SCORE

C.1 DISENTANGLING FOR REPRESENTATION INTERPRETABILITY

In this section, we argue that disentangling the sidechannel from the concepts is neither a sufficient, nor necessary condition for achieving a representation-interpretable CSM. Moreover, it may even needlessly harm the CSM's accuracy. This highlights that representation interpretability and SIS capture a distinct notion of interpretability not already addressed by disentanglement.

Zabounidis et al. (2023) propose disentangling the sidechannel from the concepts. This improves representation interpretability by preventing concept-related information from leaking into the sidechannel. However, as noted in Section 4, this is not sufficient: task-relevant information that is unrelated to the concepts is still free to appear in the sidechannel.

Importantly, disentanglement is not strictly necessary for a CSM to be representation interpretable. The simplest example is when the task predictor relies only on the concepts for each prediction. In this case, the prediction is representation interpretable regardless of what information the sidechannel encodes.

A more interesting example is the following, where disentanglement is clearly unnecessary and even hurts the model's accuracy. Suppose the dataset is such that the concepts are sufficient for solving the task. That is, they capture all task-relevant information. Consider CMR (Debot et al., 2024), a CSM where the sidechannel is a one-hot mask. The task predictor contains a set of logic rules. In default mode, the predictor uses the sidechannel to select a single rule to apply to the concepts; in bottleneck mode, it applies all rules simultaneously. For some inputs, the prediction made in default mode (via rule selection) may coincide with the prediction in bottleneck mode (via all rules), meaning the prediction is representation interpretable. However, if the sidechannel were disentangled from the concepts, rule selection could not exploit any task-relevant information (since all of it resides in the concepts). In that case, accuracy would collapse, as the rule selection would effectively become random.

C.2 Concept Completeness Score vs Sidechannel Invariance Score

In this section, we argue that the concept completeness score (Havasi et al., 2022) does not capture representation interpretability, comparing it to our SIS metric.

Havasi et al. (2022) propose a metric known as the *completeness score*, which estimates how fully concepts capture predictive information for the task. This is computed by learning a distribution $q(z|c)$ after training and relies on mutual information. While informative, it suffers from two limitations. First, mutual-information–based quantities are less intuitive for humans than accuracy-based metrics like SIS. Second, it is not a measure for representation interpretability, as it may be high even though representation interpretability is poor (see below for an in-depth discussion and a comparison between their $q(z|c)$ and our $p(z)$).

Similarly, Zhang et al. (2024) also introduce an interpretability metric, but this can only be computed for factorizable task predictors and similarly does not measure representation interpretability.

The *concept completeness score* (CCS) τ is defined by Havasi et al. (2022) as

$$\tau = \frac{I(y;c)}{I(y;c,x)} \approx \frac{H(y) + \mathbb{E}_{(x,c,y)\sim\mathcal{D}}[\log \mathbb{E}_{z\sim q(z|c)}[p(y|c,z)]]}{H(y) + \mathbb{E}_{(x,c,y)\sim\mathcal{D}}[\log \mathbb{E}_{z\sim p(z|x)}[p(y|c,z)]]} \tag{5}$$

where $H(\cdot)$ denotes entropy. This score is used to estimate how much task-relevant information is present in the concepts. Note that the numerator uses $q(z|c)$: CCS will be high whenever predictions based on z derived from c resemble those based on z derived from x. Crucially, this can happen even if the task predictor heavily relies on the sidechannel, so long as all the information in the sidechannel is also present in the concepts. Intuitively: *the concept completeness score measures how much information beyond concepts is encoded in the sidechannel; SIS measures how much the sidechannel is used.* CCS can be very high for a completely uninterpretable prediction, which is not the case for SIS. Concept-related information encoded in an uninterpretable form (e.g. embeddings) is not interpretable, since humans cannot identify what it represents, despite CCS being high. Furthermore, concepts are explicitly supervised to align with human-understandable interpretations, whereas sidechannels are not. Thus, even if a sidechannel has a simple form (e.g. a logic rule, as in DCR (Barbiero et al., 2023)) and is predicted from the same information as the concepts, we argue it remains uninterpretable. Consider the following two examples.

Consider an example with DCR. Suppose the concepts capture all task-relevant information, and the model uses five concepts $c_1, c_2, \ldots, c_5$. For an input x with ground-truth label $y = \text{True}$, assume all concepts are predicted as *True*, and the sidechannel outputs the rule $y \leftarrow c_1 \wedge c_2 \wedge \cdots \wedge c_5$. DCR's task predictor applies this rule to the predicted concepts, correctly predicting $y = \text{True}$. For this individual prediction, τ would be high, since the rule is derived from concept-related information. However, it is opaque to the human why the model used this particular rule. Why not $y \leftarrow c_1 \wedge c_2$, which would yield the same correct label, or even $y \leftarrow c_1$, or $y \leftarrow \text{True}$? In the last case, the concepts are bypassed entirely, and the prediction is made solely by the sidechannel: a neural network, hence uninterpretable. The core issue is that the sidechannel is not explicitly *aligned* with human-understandable semantics, unlike the concepts. If it were, then it would be a concept.

Consider an example with LRM. Suppose the concepts capture all task-relevant information, and the model uses some concepts and a sidechannel z that is a single neuron. These are passed to a linear layer for the task prediction. If the linear layer has learned every weight to be zero except the weight on the sidechannel, then the task prediction probability is determined entirely by the sidechannel, i.e.

by the underlying neural network that predicts z. In this case, τ will be 1, since z is derived from concept-related information. However, the interpretability the linear layer provides is completely lost. A human examining the weights will only see that the task decision (completely) depends on z, but has no way to understand how the individual concepts contribute. While the linear layer provides functional interpretability, it is useless because the model is completely not representation interpretable: the linear layer purely uses an uninterpretable representation.

Similarly, the example in Section 3.5 would also have a high completeness score.

Using $p(z)$ versus $q(z \mid c)$. One could consider defining bottleneck mode using $q(z \mid c)$ instead of $p(z)$, where the former is parameterized by a neural network. This substitution would increase expressivity: CSMs that currently act as linear classifiers ($C \to Y$) in bottleneck mode under $p(z)$ would become universal classifiers ($C \to Y$) under $q(z \mid c)$, since q is parameterized by a neural network. However, as we explained above, this added flexibility comes at the cost of interpretability. With $p(z)$, the mapping from $C \to Y$ in bottleneck mode is interpretable for functionally interpretable CSMs, whereas with $q(z \mid c)$ a neural network is put between concepts and task predictions, reducing interpretability.

D Experiments

D.1 Experimental details

Datasets. In MNIST-Addition (Manhaeve et al., 2018), each input consists of two MNIST images each representing some digit. The task is to predict the sum of the 2 digits. The concepts denote which digit is in each image. In CelebA (Liu et al., 2015), each input is the face of a celebrity, and the concepts are face attributes such as *Blond Hair* and *Wears Make-up*. As tasks, we take the concepts *Male*, *Young* and *Attractive*, dropping them from the concept set. In CUB (Welinder et al., 2010), the input is an image of a bird, the task is to classify the bird species, and concepts are bird-related features (e.g. the wing pattern).

Metrics. For CelebA, we use regular accuracy and SIS as defined in the main text. For MNIST-Addition, we use *subset* versions of these metrics due to the large imbalance and mutually exclusive nature of both concepts and tasks. For SIS, this means that an individual prediction counts as invariant only if all task outputs agree between default and bottleneck mode.

Seeds. We use seeds 1, 2 and 3.

Hard concepts. To avoid the problem of concept leakage which harms interpretability (Marconato et al., 2022), we employ hard concepts (as opposed to soft concepts) by thresholding the concept predictions at 50% before passing them to the task predictor, which is common for CBMs.

Training. We use a training objective similar to sequential training. We maximize the likelihood of our data (see Section 3.4), with as difference that we use the predicted concepts $\hat{c}$ instead of the ground truth labels for c in the task predictor $p(y|c, z)$. This makes the model more robust and aware of mistakes in the concept prediction. Importantly, we do not allow the gradient from y to pass through c (so we avoid the joint training objective some CBMs employ), as this is known to cause task leakage, harming interpretability (Mahinpei et al., 2021). For CEM, we also use its *RandInt* regularization. We always use the AdamW optimizer with learning rate 0.001 and train for 80 epochs, restoring the weights that resulted in the lowest validation loss. Our training/validation split is 9/1. For CelebA and MNIST-Addition, we use a batch size of 512. For CUB CSM training, we use a batch size of 256. The batch size of 128 stated below applies only to the separate ResNet18 finetuning stage for CUB.

Backbones. For each model, we use the same backbone (which differs between datasets). For CelebA, we train on pretrained ResNet18 embeddings (He et al., 2016), similar to Debot et al. (2024). Images are first resized to (224, 224) using bi-linear interpolation. They are then normalized per channel with means (0.485, 0.456, 0.406) and standard deviations (0.229, 0.224, 0.225). By removing the last layer of the pretrained ResNet18, using the resulting model on each image, and flattening the output, we obtain an embedding. For MNIST-Addition, we train directly on the images. The backbone is a CNN (learned jointly with the rest of the CSM) that consists of the following layers: a convolution layer with 6 output channels and kernel size 5, a max-pool layer with kernel size

and stride 2, a ReLU activation, a convolution layer with 16 output channels and kernel size 5, a max-pool layer with kernel size and stride 2, a ReLU activation, a flattening layer, a linear layer with $emb_size//2$ output features, and 3 linear layers each with $emb_size//2$ output features and ReLU activation (except the last one). The backbone is applied to each MNIST image, and the two resulting embeddings are concatenated. emb_size is a hyperparameter. If any hyperparameters are unmentioned, we use the default values.

Concrete CSMs. We will describe our implementation of the different CSMs. For each CSM, the input is first passed through the backbone before being passed to the layers we will describe in what follows. Unless explicitly mentioned otherwise, each linear layer has emb_size output features.

For CRM (Mahinpei et al., 2021), the concept predictor is a neural network consisting of 2 linear layers with ReLU activation, and a linear layer with n_C output features with sigmoid activation. The task predictor is a neural network with 3 linear layers and a linear layer with n_y output features and sigmoid activation. The sidechannel is a neural network with 2 linear layers with ReLU activation, and a linear layer, outputting an embedding. The prior $p(z)$ is denoted by a single learnable Torch embedding object with emb_size weights.

For CEM (Espinosa Zarlenga et al., 2022), the sidechannel is a neural network with 3 linear layers with $2{\cdot}c_emb_size{\cdot}n_c$ output features and ReLU activation, and a linear layer with $2{\cdot}c_emb_size{\cdot}n_c$ output features (with c_emb_size a hyperparameter). This is reshaped to $n_c, 2, c_emb_size$. The concept predictor concatenates for each concept its 2 embeddings, and applies a (different) linear layer to the resulting embedding with 1 output feature and sigmoid activation. This is the concept prediction. The task predictor uses the concept predictions to mix for each concept its 2 embeddings, and concatenates the resulting embeddings. The result is passed through 3 linear layers with ReLU activation, and a linear layer with n_y output features and sigmoid activation. The prior $p(z)$ is denoted by a single learnable Torch embedding object with $2 \cdot c_emb_size \cdot n_c$ weights.

For DCR (Barbiero et al., 2023), the sidechannel is a neural network with 3 linear layers with $2 \cdot c_emb_size \cdot n_c$ output features and ReLU activation, and a linear layer with $2 \cdot c_emb_size \cdot n_c$ output features (with c_emb_size a hyperparameter). This is reshaped to $n_c, 2, c_emb_size$. The concept predictor concatenates for each concept its 2 embeddings, and applies a (different) linear layer to the resulting embedding with 1 output feature and sigmoid activation. This is the concept prediction. The task predictor uses the concept predictions to mix for each concept its 2 embeddings. Each resulting *concept embedding* is passed through a (different) neural network consisting of 3 linear layers with c_emb_size output features and ReLU activation, and a linear layer with $2 \cdot n_y$ output features with sigmoid activation. These should be interpreted as, for each concept, its polarity and relevance for each task, as defined by DCR. These are used together with the concept predictions to infer the task prediction. We employ DCR's logic formula (see Barbiero et al. (2023)) using the product t-norm. The prior $p(z)$ is denoted by a single learnable Torch embedding object with $2 \cdot c_emb_size \cdot n_c$ weights. To avoid needing to finetune DCR's *temperature* hyperparameter, which a human user would do to find a preferred rule parsimony, we modelled DCR's polarity and relevance with a sigmoid activation instead of their rescaled softmax activation.

For CMR (Debot et al., 2024), the concept predictor is a neural network consisting of 2 linear layers with ReLU activation, and a linear layer with n_C output features with sigmoid activation. The sidechannel is a neural network with 3 linear layers with ReLU activation, and a linear layer with $n_r \cdot n_y$ output features, where n_r is CMR's allowed number of rules to learn (hyperparameter). This is reshaped to (n_y, n_r) with a softmax on the last dimension (as this is a categorical random variable). The task predictor consists of a Torch embedding object with shape $n_y, n_r, rule_emb_size$ weights, with $rule_emb_size$ a hyperparameter. This is CMR's rulebook in its latent representation, which is decoded into explicit logic rules by using a neural network on each rule embedding with 3 linear layers with ReLU activation and a linear layer with $3 \cdot n_c$ output features with softmax activation. The task predictor then uses CMR's inference formula (see Debot et al. (2024)) to derive the task prediction from the categorical distribution over rules (given by the sidechannel) and the learned set of rules. In bottleneck mode, CMR applies the entire set of rules to the concepts, each making a y prediction. The final prediction is the logical OR of the individual predictions. CMR's hyperparameter for its prototype regularization is set to 0 for the same reason as why we changed DCR's role and relevance activation, as this hyperparameter would be used by the human to find a preferred rule parsimony.

For LRM, we use the same setup as for CRM, with as difference that the task predictor is a linear layer with n_y output features and sigmoid activation.

For SIS regularization, we always use KL divergence.

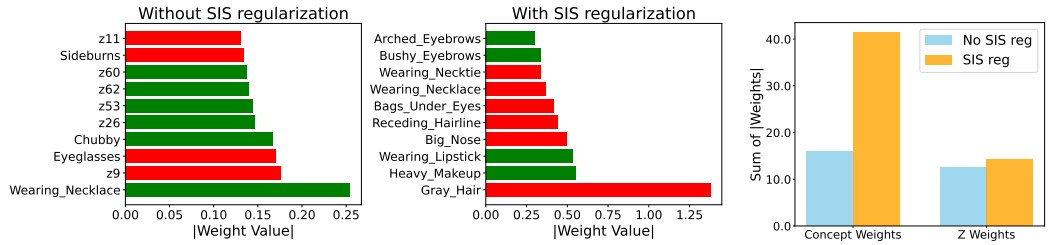
CMR*. We define CMR* as an adaptation of CMR by equipping it with rules learned by a decision tree. Specifically, we extract the rules decision trees learned on ground truth (c, y) pairs predicting positive classes. We start with decision trees with a maximum depth of 1, increasing it until the decision tree has more than 17 such rules, or depth exceeds 40. Then, we take the rules from the tree with the highest validation accuracy. We inject these rules into CMR, and allow it to learn 3 additional rules on its own. This is done through rule interventions, which CMR supports (see Debot et al. (2024)).

For obtaining Figure 3, we perform a **grid search** where emb_size is taken from $\{64, 128, 256\}$ and the weight of the SIS regularization is taken from $\{0, 0.0001, 0.001, 0.01, 0.05, 0.1, 0.5, 1.0, 1.5, 2.0, 3.0, 4.0, 5.0\}$. For CEM and DCR, we use $p_{RandInt} = 0.05$. **For obtaining Figure 4**, we have 2 phases. In the first phase, we perform the same grid search except that we do not employ SIS regularization. For each CSM, we take the most accurate configuration on the validation set. Then, in the second phase, for each CSM, we train this configuration with different values of the SIS regularization weight, taken from $\{0, 0.0001, 0.001, 0.01, 0.05, 0.1, 0.5, 1.0, 1.5, 2.0, 3.0, 4.0, 5.0\}$. For each such trained model, we compute its intervenability curve by (1) generating a random concept intervention order, (2) intervening on increasingly more concepts following this order, (3) each time computing the accuracy on the tasks after the intervention. **For obtaining Figure 5**, we take the most accurate LRM on the validation set (taken from the first mentioned grid search) with and without SIS regularization and inspect the learned weights. **For obtaining the CRM* and CRM** results from Table 1**, we train a *dropout* version inspired by Kalampalikis et al. (2025) and a *detach* version inspired by Shang et al. (2024). For the dropout version, we extend the grid by considering values for $p_{dropout}$ from $\{0.0, 0.2, 0.4, 0.6, 0.8, 1.0\}$. For more details of these two versions, see below. **For obtaining Figure 10**, we first finetune a ResNet18 on CUB, where we add a linear layer with a softmax for predicting the classes and a linear layer with a sigmoid for predicting the concepts. We use a learning rate of 0.01 (SGD optimizer with momentum of 0.9), batch size 128 and train for 200 epochs. This is similar to the configuration many other CBM works use (Espinosa Zarlenga et al., 2022). We then drop the classification layers and run the CUB images through the ResNet18 to obtain image embeddings, which we use to train the CSMs on (instead of on the images). We use the following hyperparameter grid: SIS weight within $\{0, 0.0001, 0.001, 0.01, 0.05, 0.1, 0.5, 1.0, 1.5, 2.0, 3.0, 4.0, 5.0, 7.0, 10.0, 12.0, 20.0, 50.0\}$, emb_size 512, sidechannel number of hidden layers between 0 and 2, concept predictor number of hidden layers either 0 or 1, task predictor number of hidden layers between 0 and 2. For Table 1, the reference point is $(a, 0)$ for each dataset, where a is the majority-classifier accuracy under that dataset's task metric.

Pareto-efficient points selection. For each dataset, architecture, and seed, we evaluate all hyperparameter configurations on the validation set. A configuration is considered Pareto-efficient if no other configuration achieves both validation accuracy at least as high and validation SIS at least as high, with a strict improvement in at least one metric. Pareto filtering is performed separately for each architecture. The configurations selected in this manner are then evaluated on the held-out test set, and their test accuracy and SIS are displayed in the figures.

Detach version (CRM)**. We adapt CRM's architecture to resemble Shang et al. (2024). Concretely, we redefine the task predictor as $y = f(c) + g(z)$, where y are the output logits. For g, we use a linear layer with n_y output features. For f, we use a neural network consisting of 3 linear layers with ReLU activation, and a linear layer with n_y output features. The model is trained by optimizing (1) a cross entropy between $f(c)$ and the label, and (2) a cross entropy between $f(c).detach() + g(z)$ and the label. The first objective tries to maximize concept usage, while the second one trains the sidechannel.

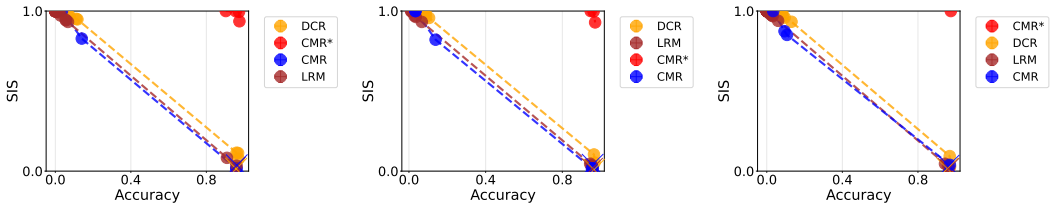
Dropout version (CRM*). During training, for each element in each batch, we set the entire sidechannel to 0 with probability $p_{dropout}$, which is a hyperparameter. This encourages the task predictor to rely on the concepts when the sidechannel is dropped out.



(a) Top 10 largest weights. Bars represent absolute weight magnitudes, with green and red indicating positive and negative weights, respectively. Weights corresponding to the sidechannel's neurons are denoted with a 'z'.

(b) Total absolute weight magnitudes assigned to concepts and sidechannel neurons.

Figure 5: Inspection of LRM's linear layer for predicting the task 'Young' for CelebA, comparing LRMs trained with and without SIS regularization.



(a) Seed 1

(b) Seed 2

(c) Seed 3

Figure 6: Accuracy vs SIS trade-off in functionally interpretable CSMs, including CMR*, on MNIST-Add. Different points are different hyperparameter configurations, keeping only pareto-efficient points. Crosses denote the most accurate configuration trained without SIS regularization.

D.2 Additional results

With a different rule learner, CMR can achieve high accuracy when allowed to learn enough rules (Figure 6). CMR* achieves near-perfect accuracy and representation interpretability on MNIST-Addition.

Figures 7 and 8 give additional accuracy-interpretability trade-off results for seeds 2 and 3. Figure 9 shows some additional intervenability curves, which also show that SIS regularization improves intervenability (similar to Figure 4 in the main text).

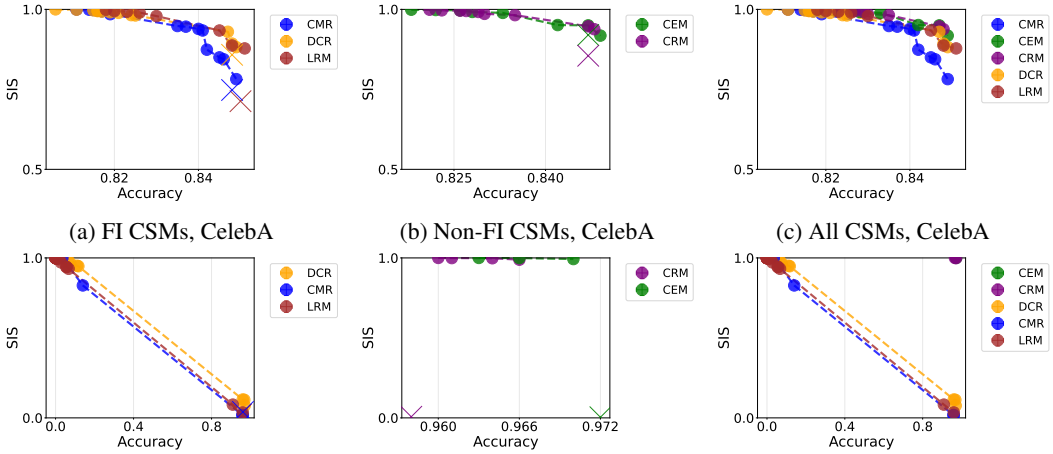
On CUB, we only used CRM, LRM and CEM because CMR and DCR do not naturally support multiclass classification. Figure 10 shows their accuracy-interpretability trade-off. Figure 11 gives an alternative view similar to the pareto curves in the main text.

Figure 12's concept reliance is computed as follows:

$$\frac{\sum_{j=1}^{n_c} \mathrm{SD}_{\mathrm{train}}(c_j) \sum_{t=1}^{n_y} \left|W_{t,j}^{(c)}\right|}{\sum_{j=1}^{n_c} \mathrm{SD}_{\mathrm{train}}(c_j) \sum_{t=1}^{n_y} \left|W_{t,j}^{(c)}\right| + \sum_{k=1}^{n_z} \mathrm{SD}_{\mathrm{train}}(z_k) \sum_{t=1}^{n_y} \left|W_{t,k}^{(z)}\right|}.$$

Here, c_j denotes predicted concept j, z_k denotes sidechannel feature k, and $W^{(c)}$ and $W^{(z)}$ are their respective task-layer weights. For each trained checkpoint, each feature's absolute task-layer weight is multiplied by its training-set standard deviation (computed on the predicted concept and sidechannel activations), measuring its contribution under a one-standard-deviation change in that feature. Concept reliance is therefore the fraction of total scale-adjusted task-layer weight attributable to concepts. Each point in the figure is the mean concept-reliance score across the trained models at that value of β. The uncertainty denotes the standard-deviation across these scores.

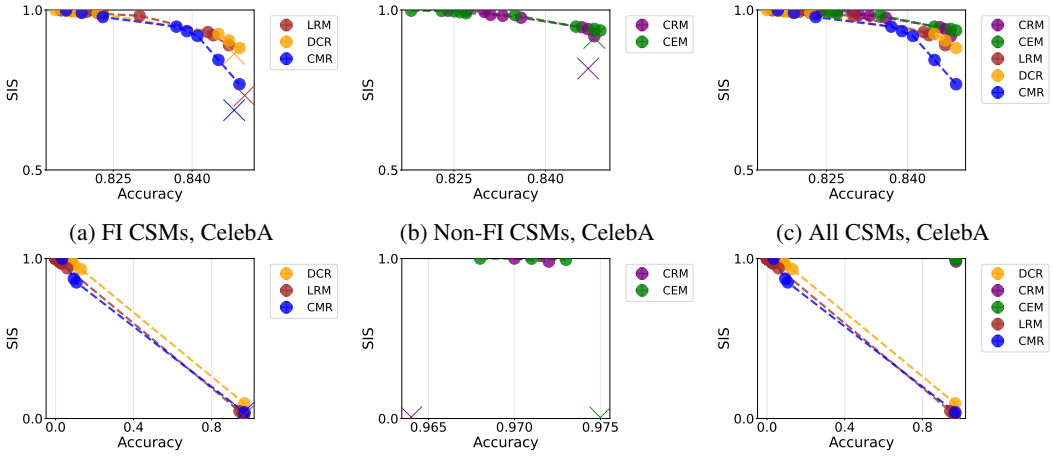
We ran our experiments on a machine with an Nvidia L40S 48 GB GPU with an AMD EPYC 9334 CPU and 256 GB RAM. Small deviations in results are due to the different machine settings. We



(a) FI CSMs, CelebA (b) Non-FI CSMs, CelebA (c) All CSMs, CelebA

(d) FI CSMs, MNIST-Add. (e) Non-FI CSMs, MNIST-Add. (f) All CSMs, MNIST-Add.

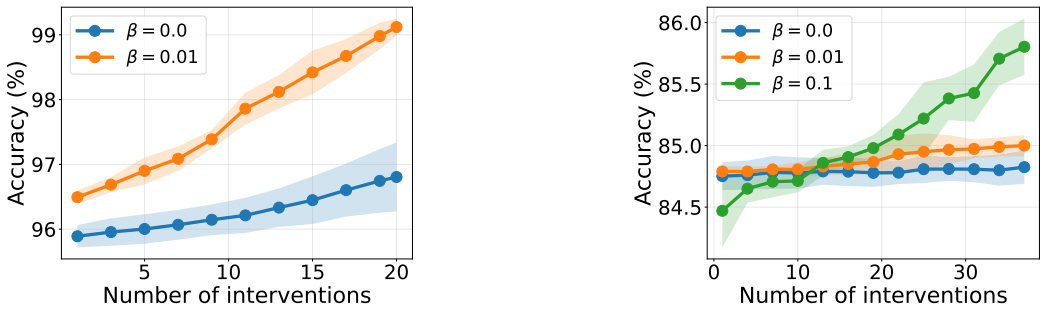
Figure 7: Accuracy vs SIS trade-off in CSMs (seed 2). Different points are different hyperparameter configurations, keeping only pareto-efficient points. Crosses denote the most accurate configuration trained without SIS regularization (excluded from (c) and (f)). "(Non)-FI" denotes the figure only contains (non)-functionally interpretable CSMs.



(a) FI CSMs, CelebA (b) Non-FI CSMs, CelebA (c) All CSMs, CelebA

(d) FI CSMs, MNIST-Add. (e) Non-FI CSMs, MNIST-Add. (f) All CSMs, MNIST-Add.

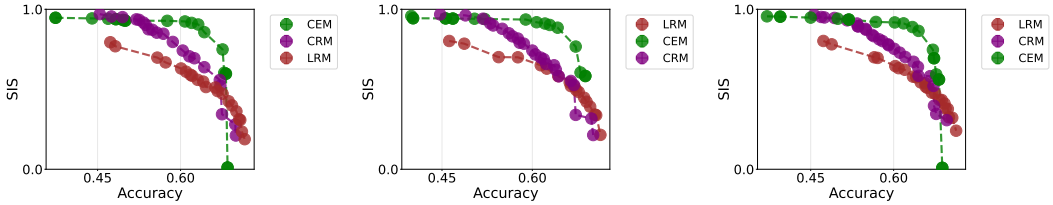
Figure 8: Accuracy vs SIS trade-off in CSMs (seed 3). Different points are different hyperparameter configurations, keeping only pareto-efficient points. Crosses denote the most accurate configuration trained without SIS regularization (excluded from (c) and (f)). "(Non)-FI" denotes the figure only contains (non)-functionally interpretable CSMs.



(a) CMR* on MNIST-Addition

(b) CEM on CelebA

Figure 9: Intervenability in CSMs with ($\beta > 0$) and without ($\beta = 0$) SIS regularization for different regularization weights β. The y-axis denotes accuracy after intervening on a number of concepts denoted by the x-axis.



(a) Seed 1

(b) Seed 2

(c) Seed 3

Figure 10: Accuracy vs SIS trade-off on CUB. Different points are different hyperparameter configurations, keeping only pareto-efficient points. Crosses denote the most accurate configuration trained without SIS regularization.

estimate that it takes 180h to run all experiments, and that an additional 180h was spent on preliminary experiments.

E Additional Theoretical Results

We first mention notation used throughout this section. Let (X, C, Z, Y) be input, concept, sidechannel, and target task random variables taking values in $\mathcal{X}$, $\mathcal{C}$, $\mathcal{Z}$, and the finite label set $\mathcal{Y}$, respectively. For an input-independent prior $q \in \Delta(\mathcal{Z})$, the default and bottleneck predictive distributions at input $x \in \mathcal{X}$ are $p(Y \mid x)$ and $p(\bar{Y} \mid x, q)$, where $\bar{Y}$ denotes the task output random variable in bottleneck mode under q. We write the induced classifiers as

$$h(x) = \arg\max_{y \in \mathcal{Y}} p(Y = y \mid x), \qquad \bar{h}_q(x) = \arg\max_{y \in \mathcal{Y}} p(\bar{Y} = y \mid x, q),$$

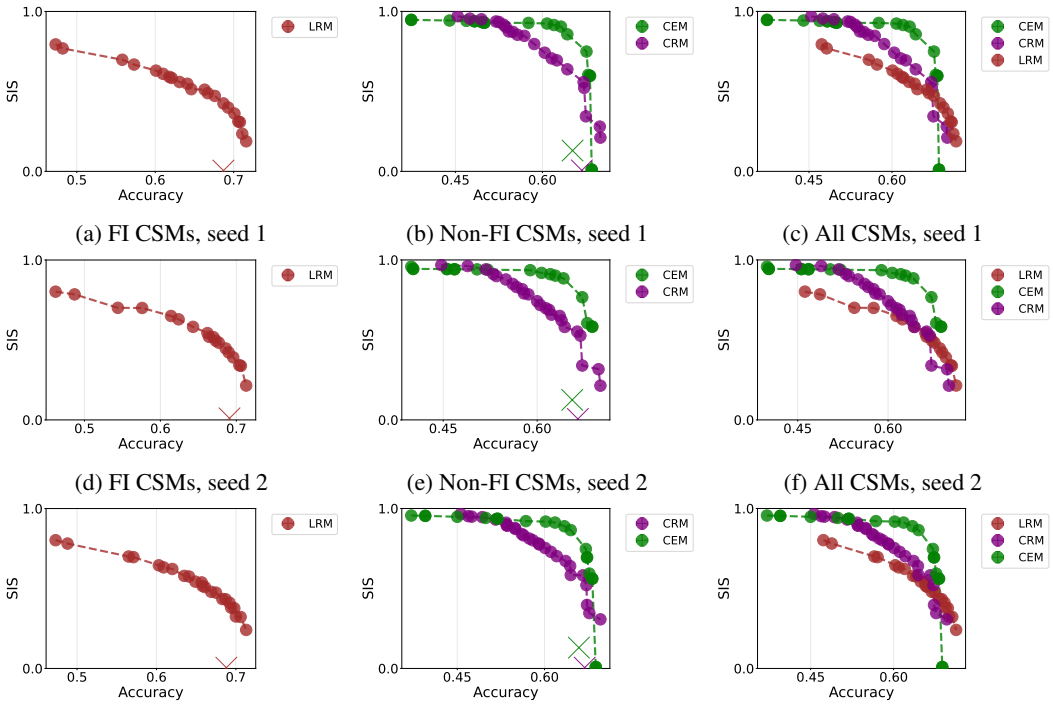
so that $h(x) = y_x$ and $\bar{h}_q(x) = \bar{y}_x^q$ in the notation of Section 3.4. For a classifier f, let $R(f) = p(f(X) \neq \hat{Y})$ denote its zero–one risk, with $\hat{Y}$ the ground-truth. We write $H(\cdot)$ for Shannon entropy and $I(\cdot\,;\cdot)$ for mutual information.

For a fixed $q \in \Delta(\mathcal{Z})$ we write

$$\mathrm{SIS}_q \;:=\; \mathbb{E}_{x \sim p(X)}\big[\mathbb{1}[y_x = \bar{y}_x^q]\big] \;=\; \mathbb{E}_{x \sim p(X)}\big[\mathbb{1}[h(x) = \bar{h}_q(x)]\big], \quad \text{so that} \quad \mathrm{SIS} = \sup_{q \in \Delta(\mathcal{Z})} \mathrm{SIS}_q.$$

The supremum need not be attained. All results below are proved uniformly in q and therefore require no attainment or compactness assumption. Ties in the $\arg\max$ defining y_x and $\bar{y}_x^q$ are broken by a fixed rule (lowest class index), so both are well-defined.

Results below are stated for a fixed q in terms of SIS_q. Lower bounds on SIS_q at any fixed q transfer to SIS, since $\mathrm{SIS} \geq \mathrm{SIS}_q$. Bounds that use $1 - \mathrm{SIS}_q$ to control a difference of risks are uniform in q and are lifted to SIS by minimizing the left-hand side over q (Theorems E.2 and E.3).

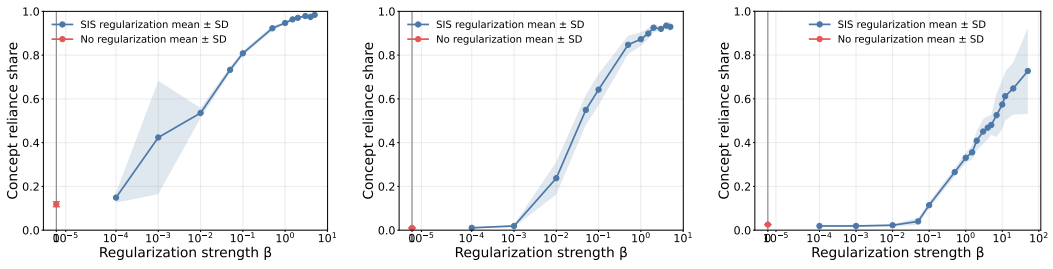


(a) FI CSMs, seed 1 (b) Non-FI CSMs, seed 1 (c) All CSMs, seed 1

(d) FI CSMs, seed 2 (e) Non-FI CSMs, seed 2 (f) All CSMs, seed 2

(g) FI CSMs, seed 3 (h) Non-FI CSMs, seed 3 (i) All CSMs, seed 3

Figure 11: Accuracy vs SIS trade-off in CSMs for CUB (all seeds). Different points are different hyperparameter configurations, keeping only pareto-efficient points. Crosses denote the most accurate configuration trained without SIS regularization (excluded from (c), (f) and (i)). "(Non)-FI" denotes the figure only contains (non)-functionally interpretable CSMs.



(a) CelebA (b) MNIST-Add (c) CUB

Figure 12: Concept reliance as a function of the SIS-regularization strength β for trained LRMs. Lines show the mean across runs and shaded regions indicate standard deviation. For models trained without SIS regularization, only the highest accuracy run per seed is retained.

E.1 Unavoidable Reliance from insufficiency and inexpressivity

We first prove a lemma to help in proving future theorems.

Lemma E.1. *Let $q \in \Delta(\mathcal{Z})$ be arbitrary and let $\bar{h}_q$ be the bottleneck classifier it induces. Then*

$$|R(h) - R(\bar{h}_q)| \;\le\; 1 - \mathrm{SIS}_q.$$

Equivalently, if $A(f) = 1 - R(f)$ is accuracy, then

$$A(h) - \big(1 - \mathrm{SIS}_q\big) \;\le\; A(\bar{h}_q) \;\le\; A(h) + \big(1 - \mathrm{SIS}_q\big).$$

More generally, let $B > 0$ and let $\ell : \mathcal{Y} \times \mathcal{Y} \to [0, B]$ be a bounded loss. Then the absolute difference between the two expected losses is at most $B\,\big(1 - \mathrm{SIS}_q\big)$.

Proof. Pointwise,

$$\big|\mathbb{1}[h(x) \neq y] - \mathbb{1}[\bar{h}_q(x) \neq y]\big| \;\le\; \mathbb{1}[h(x) \neq \bar{h}_q(x)].$$

Taking expectations and using $|\mathbb{E}[A] - \mathbb{E}[B]| \le \mathbb{E}|A - B|$ proves the zero–one statement. For the bounded-loss version, the pointwise loss difference is zero whenever $h(x) = \bar{h}_q(x)$ and at most B otherwise. □

Below, we formalize the notions introduced in Section 3.5: a term belonging to missing task information in the concepts, and a term representing the inability of e.g. a linear concept-to-task predictor to express a concept-sufficient target such as XOR. If the bottleneck representation (the concepts) or its predictor class has unavoidable excess error, a sufficiently accurate default model cannot have arbitrarily high SIS.

Theorem E.2 (Unavoidable reliance from insufficiency and inexpressivity)**.** *Let $\mathcal{H}_B$ be a subclass of all classifiers measurable with respect to the concept representation C (classifiers in bottleneck mode). Let $h(x) = \arg\max_{y \in \mathcal{Y}} p(y \mid x)$ and $\bar{h}(x) = \arg\max_{y \in \mathcal{Y}} p(Y = y \mid x)$, and let $R(f) = p(f(X) \neq Y)$ denote the zero-one risk for a classifier f. Define*

$$R^*_B := \inf_{g \in \mathcal{H}_B} R(g), \qquad R^*_C := \inf_{g : \mathcal{C} \to \mathcal{Y}} p(g(C) \neq Y), \qquad R^*_X := \inf_{g : \mathcal{X} \to \mathcal{Y}} p(g(X) \neq Y).$$

If the default classifier h is Bayes-optimal, then every $\bar{h} \in \mathcal{H}_B$ satisfies

$$\mathrm{SIS} \le 1 - \underbrace{(R^*_C - R^*_X)}_{\textit{concept insufficiency}} - \underbrace{(R^*_B - R^*_C)}_{\textit{bottleneck-class approximation error}} = 1 - (R^*_B - R^*_X).$$

Proof. Fix any $q \in \Delta(\mathcal{Z})$. Lemma E.1 gives $1 - \mathrm{SIS}_q \ge R(\bar{h}_q) - R(h)$. Since $\bar{h}_q \in \mathcal{H}_B$ we have $R(\bar{h}_q) \ge R^*_B$, and since h is Bayes-optimal $R(h) = R^*_X$, so $1 - \mathrm{SIS}_q \ge R^*_B - R^*_X$ for every q. The right-hand side does not depend on q, so the inequality holds with the left-hand side minimized over q; as $\inf_q(1 - \mathrm{SIS}_q) = 1 - \sup_q \mathrm{SIS}_q = 1 - \mathrm{SIS}$, we obtain $1 - \mathrm{SIS} \ge R^*_B - R^*_X$. Because $\mathcal{H}_B$ is a subclass of the C-measurable classifiers, $R^*_X \le R^*_C \le R^*_B$. Consequently, $R^*_B - R^*_X = (R^*_C - R^*_X) + (R^*_B - R^*_C)$. □

We can generalize Theorem E.2 to include non-Bayes-optimal classifiers by including an excess risk above the Bayes risk.

Theorem E.3 (Unavoidable reliance for any classifier)**.** *Let $h : \mathcal{X} \to \mathcal{Y}$ be any default classifier and define its excess risk above the Bayes risk by*

$$\xi(h) := R(h) - R^*_X \ge 0.$$

Under the assumptions and notation of Theorem E.2, every $\bar{h} \in \mathcal{H}_B$ satisfies

$$1 - \mathrm{SIS} \ge \max\left(\underbrace{R^*_C - R^*_X}_{\textit{concept insufficiency}} + \underbrace{R^*_B - R^*_C}_{\textit{bottleneck-class approximation error}} - \underbrace{\xi(h)}_{\textit{default excess risk}}, 0 \right).$$

Equivalently,

$$\mathrm{SIS} \le 1 - \max\left((R^*_C - R^*_X) + (R^*_B - R^*_C) - \xi(h), 0\right).$$

Proof. Fix any $q \in \Delta(\mathcal{Z})$. Lemma E.1 gives $1 - \mathrm{SIS}_q \geq R(\bar{h}_q) - R(h) \geq R^*_B - R(h)$. Substituting $\xi(h) = R(h) - R^*_X$,

$$R^*_B - R(h) = (R^*_C - R^*_X) + (R^*_B - R^*_C) - \xi(h).$$

The right-hand side does not depend on q, so minimizing the left-hand side over q gives $1 - \mathrm{SIS} \geq (R^*_C - R^*_X) + (R^*_B - R^*_C) - \xi(h)$, and $1 - \mathrm{SIS} \geq 0$ allows the maximum with 0. Both displayed forms follow. □

This shows that Bayes optimality is not needed. It merely gives the strongest ceiling by setting $\xi(h) = 0$. For an imperfect default model, its excess risk weakens the bound, as bottleneck mode can agree with a poor default mode without matching the best performance available from X.

E.2 Proof of Theorem 3.1

Proof. Condition on all training and validation data, all algorithmic randomness, and hence the realized pair $(h_{\mathrm{sel}}, \bar{h}_{\hat{q}})$. Independence of S_{test} makes the n agreement indicators i.i.d. Bernoulli variables with conditional mean $\mathrm{SIS}_{\hat{q}}$. The one-sided Hoeffding inequality gives

$$p\Big(\widehat{\mathrm{SIS}}_{\mathrm{test}} - \mathrm{SIS}_{\hat{q}} > t \;\Big|\; h_{\mathrm{sel}}, \bar{h}_{\hat{q}}\Big) \;\leq\; e^{-2nt^2}.$$

Setting the right-hand side equal to δ and solving for t gives $t = \sqrt{\log(1/\delta)/2n}$. Since the fitted prior $p_\gamma(z)$ is one element of $\Delta(\mathcal{Z})$, Definition 3.3 gives $\mathrm{SIS} \geq \mathrm{SIS}_{\hat{q}}$, and therefore $\mathrm{SIS} \geq \widehat{\mathrm{SIS}}_{\mathrm{test}} - t$. Because the conditional guarantee holds for every realized selected pair, averaging over the training and selection randomness proves the unconditional statement. □

E.3 Link between SIS regularization and SIS

Theorem E.4 (Conversion from KL to SIS)**.** *Let $\mathcal{Y}$ be a finite set of classes. Define the probability margin of the default classifier by*

$$\Delta(x) \;:=\; p\big(Y = h(x) \mid X = x\big) \;-\; \max_{y \in \mathcal{Y} \setminus \{h(x)\}} p\big(Y = y \mid X = x\big),$$

which is the difference in probability between the two most likely classes. Then, for every $\tau > 0$ and every $q \in \Delta(\mathcal{Z})$,

$$\mathrm{SIS}_q \;\geq\; 1 - \mathbb{E}_{x \sim p(X)}\big[\mathbb{1}[\Delta(x) \leq \tau]\big] \;-\; \frac{\sqrt{2\,\mathbb{E}_{x \sim p(X)}\big[\mathrm{KL}\big(p(Y \mid x) \,\big\|\, p(\bar{Y} \mid x, q)\big)\big]}}{\tau},$$

where $\mathbb{1}[\Delta(x) \leq \tau]$ equals one when $\Delta(x) \leq \tau$ and zero otherwise. In particular $\mathrm{SIS} \geq \mathrm{SIS}_q$*, so the right-hand side also lower-bounds* SIS*.*

Proof. Fix a prior $q \in \Delta(\mathcal{Z})$ and an input $x \in \mathcal{X}$, and write

$$i := h(x), \qquad j := \bar{h}_q(x).$$

Suppose that the two classifiers disagree, so that $i \neq j$. Because j maximizes the bottleneck output probabilities,

$$p(\bar{Y} = j \mid X = x, q) \;\geq\; p(\bar{Y} = i \mid X = x, q).$$

Moreover, because $j \neq i$, the definition of the default margin gives

$$\Delta(x) \;\leq\; p(Y = i \mid X = x) - p(Y = j \mid X = x).$$

Combining these observations yields

$$\begin{aligned}
\Delta(x) \;&\leq\; p(Y = i \mid X = x) - p(Y = j \mid X = x) \\
&\leq\; \big|p(Y = i \mid X = x) - p(\bar{Y} = i \mid X = x, q)\big| \\
&\quad + \big|p(Y = j \mid X = x) - p(\bar{Y} = j \mid X = x, q)\big| \\
&\leq\; \big\|p(Y \mid X = x) - p(\bar{Y} \mid X = x, q)\big\|_1.
\end{aligned}$$

Pinsker's inequality in ℓ_1 form therefore gives

$$\Delta(x) \;\le\; \sqrt{2\,\mathrm{KL}\big(p(Y \mid X=x)\,\big\|\,p(\bar{Y} \mid X=x,q)\big)}.$$

For brevity, define

$$D_q(x) \;:=\; \mathrm{KL}\big(p(Y \mid X=x)\,\big\|\,p(\bar{Y} \mid X=x,q)\big).$$

The preceding argument shows that, on an input satisfying $\Delta(x) > \tau$, disagreement implies $\sqrt{2D_q(x)} > \tau$. Consequently, the disagreement indicator satisfies the pointwise bound

$$\mathbb{1}[h(x) \neq \bar{h}_q(x)] \;\le\; \mathbb{1}[\Delta(x) \le \tau] + \frac{\sqrt{2D_q(x)}}{\tau}.$$

Indeed, if $\Delta(x) \le \tau$ the first term equals one; if $\Delta(x) > \tau$ the left-hand side is zero unless the classifiers disagree, in which case the second term exceeds one. Taking expectations over $X \sim p(X)$ gives

$$1{-}\mathrm{SIS}_q \;=\; \mathbb{E}_{X\sim p(X)}\big[\mathbb{1}[h(X) \neq \bar{h}_q(X)]\big] \;\le\; \mathbb{E}_{X\sim p(X)}\big[\mathbb{1}[\Delta(X) \le \tau]\big] + \frac{\sqrt{2}}{\tau}\,\mathbb{E}_{X\sim p(X)}\big[\sqrt{D_q(X)}\big].$$

Because the square-root function is concave, Jensen's inequality gives

$$\mathbb{E}_{X\sim p(X)}\big[\sqrt{D_q(X)}\big] \;\le\; \sqrt{\mathbb{E}_{X\sim p(X)}\big[D_q(X)\big]}.$$

Substituting this inequality into the preceding bound and rearranging proves the first claim. The second follows from $\mathrm{SIS} = \sup_{q'} \mathrm{SIS}_{q'} \ge \mathrm{SIS}_q$. □

The theorem separates two possible causes of disagreement between the default and bottleneck classifiers. The first term measures how frequently the default classifier has a small probability margin and is therefore sensitive to small changes in its output probabilities. The second term measures the average KL divergence between the default and bottleneck output distributions, which is exactly the quantity SIS regularization penalizes. Thus, a high SIS_q (and hence a high SIS) is guaranteed when low-margin inputs are rare and the average output-space KL divergence is small.

E.4 The sidechannel prior and information contraction

We will show that using the marginal for $p(z)$ is a sensible choice but suboptimal compared to a learnable prior.

Theorem E.5 (The marginal is the KL-optimal input-independent prior)**.** *For every input-independent probability distribution q on $\mathcal{Z}$ such that $p(Z)$ is absolutely continuous with respect to $q(Z)$,*

$$\mathbb{E}_{x\sim p(X)}\mathrm{KL}(p(Z \mid X=x)\|q(Z)) = I(X;Z) + \mathrm{KL}(p(Z)\|q(Z)).$$

Therefore $q(Z) = p(Z)$ minimizes the expected KL discrepancy and the minimum value is $I(X;Z)$.

Proof. Insert and subtract $\log p(z)$ inside the definition:

$$\begin{aligned}
&\mathbb{E}_{x\sim p(X)}\mathbb{E}_{z\sim p(Z|X=x)}\left[\log\frac{p(z \mid x)}{q(z)}\right] \\
&\quad= \mathbb{E}_{x\sim p(X)}\mathbb{E}_{z\sim p(Z|X=x)}\left[\log\frac{p(z \mid x)}{p(z)}\right] + \mathbb{E}_{z\sim p_Z}\left[\log\frac{p(z)}{q(z)}\right] \\
&\quad= I(X;Z) + \mathrm{KL}(p(Z)\|q(Z)).
\end{aligned}$$

Nonnegativity of KL proves optimality. □

Remark. The hypothesis requires $p(Z \mid X=x)$ to be absolutely continuous with respect to q. For CSMs whose sidechannel is a delta distribution over a continuous space (CRM, CEM), this fails for a.e. x and both sides of the identity are infinite. Theorem E.5 should therefore be read as applying to CSMs with discrete $\mathcal{Z}$ (e.g. CMR's rule selection). This does not affect what we compute: SIS regularization penalizes a divergence in output space, not in $\mathcal{Z}$.

This shows the marginal prior is meaningful: of all baselines that are forbidden to depend on x, $p(Z)$ changes the latent sidechannel distribution as little as possible in expected KL. The unavoidable residual is exactly $I(X;Z)$, the amount of input information carried by the sidechannel.

The following theorem provides a link between regularizing the output space (as with SIS regularization) and directly regularizing the latent variable Z.

Theorem E.6 (Link between latent and output regularization)**.** *Fix $x \in \mathcal{X}$, and let q be an input-independent probability distribution on $\mathcal{Z}$. Let $p(Y \mid x)$ be the default output distribution obtained using $p(Z \mid x)$, and let $p(\bar{Y} \mid x)$ be the bottleneck output distribution obtained by replacing $p(Z \mid x)$ with $q(Z)$, while keeping the concept and task predictors fixed. Then*

$$\mathrm{KL}\big(p(Y \mid x) \,\big\|\, p(\bar{Y} \mid x)\big) \le \mathrm{KL}\big(p(Z \mid x) \,\big\|\, q(Z)\big).$$

In particular, choosing $q(Z) = p(Z)$ gives

$$\mathbb{E}_{x\sim p(X)}\left[\mathrm{KL}\big(p(Y \mid x) \,\big\|\, p(\bar{Y} \mid x)\big)\right] \le I(X;Z).$$

Proof. For fixed x, define the transition kernel

$$K_x(y \mid z) = \sum_c p(c \mid x)\, p(y \mid c, z).$$

The default and bottleneck output distributions are obtained by passing $p(Z \mid x)$ and $q(Z)$, respectively, through the same kernel K_x. The data-processing inequality therefore gives

$$\mathrm{KL}\big(p(Y \mid x) \,\big\|\, p(\bar{Y} \mid x)\big) \le \mathrm{KL}\big(p(Z \mid x) \,\big\|\, q(Z)\big).$$

Averaging over x and choosing $q(Z) = p(Z)$ gives the second result. □

Remark. The first inequality of Theorem E.6 follows from the data-processing inequality and may hold even when the upper bound is infinite. In our models, the output-space KL divergence is finite because the finite label space has strictly positive sigmoid or softmax probabilities. However, the latent-space KL divergence may be infinite for deterministic continuous sidechannels such as those of CRM and CEM. The latent bound is therefore most informative for discrete sidechannels, such as CMR.

Replacing $p(Z \mid x)$ with $q(Z)$ can only become less distinguishable after the concept and task predictors map it to output probabilities. Hence, small latent KL is sufficient for small output discrepancy. Importantly, the converse need not hold: Z can carry much input information that the task predictor ignores, in which case the latent KL is large while the output discrepancy is zero. This is the reason why we regularize in output space rather than on Z. Our criterion concerns whether the sidechannel changes the prediction, not whether it carries information; constraining Z is therefore strictly stronger than necessary, and removes capacity that the model may never use at the decision level. Theorem E.6 makes this precise in one direction, and Appendix B shows a practical consequence: some CSMs are more expressive in bottleneck mode under SIS regularization with a learnable prior than they would be under a latent constraint. The output-space quantity is also the one that remains well-defined for CSMs with delta-parameterized sidechannels (see the remark following Theorem E.5).

F Code and licenses

Our code will be made publicly available upon acceptance under the Apache license, Version 2.0. We used Python 3.10.12 and the following libraries: PyTorch v2.5.1 (BSD license) (Paszke et al., 2019), PyTorch-Lightning v2.5.0 (Apache license 2.0), scikit-learn v1.5.2 (BSD license) (Pedregosa et al., 2011), PyC v0.0.11 (Apache license 2.0). We used CUDA v12.7 and plots were made using Matplotlib (BSD license). The CelebA dataset is available for non-commercial research purposes only[4] and MNIST is available on the web with the CC BY-SA 3.0 DEED license. CUB is available under the MIT license.

[4] https://mmlab.ie.cuhk.edu.hk/projects/CelebA.html