GRADIENT-CANCELLED CONCEPT-BASED MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Concept-based models are designed to support human intervention: an expert can correct an incorrect concept at test time and improve task prediction, a capability particularly important in high-stakes domains. Joint training typically achieves the strongest task performance, but this can come at the cost of information leakage, whereby concept representations encode task-relevant information not captured by their intended semantic meaning. We provide an information-theoretic characterisation of task-driven leakage in joint training: under task-head convergence, the encoder update contains an explicit mutual-information maximisation term that encourages concept representations to capture task-relevant information beyond the annotated concepts. We introduce **Gradient-Cancelled Concept-Based Models**, which use a Gradient Reversal Layer to cancel the task-driven encoder gradient, leaving the concept encoder to learn from concept supervision alone while training the task head concurrently. Our model requires a simple training-time gradient modification and applies to any concept-based model where task gradients reach a shared encoder via concept representations. Our experiments show that Gradient Cancelled (GC) training results in significant task improvements upon intervention over vanilla concept-based models, while maintaining strong concept and task baseline performance, making these models better suited to in the human-in-the-loop settings they were originally designed for.

1 INTRODUCTION

Concept-based models (CMs) were introduced as a family of models that are both interpretable-by-design and intervenable. These features are particularly desirable in high risk settings like medicine, where human oversight is often considered essential. A clinician who disagrees with a predicted concept can correct it and see the prediction change, a capability no opaque classifier offers. CMs first map inputs to a set of meaningful intermediate attributes or concepts, and then base the final prediction on these concepts. If we were interested in cancer prognosis, concepts might include *'tumour size'* or *'proliferation rate'*. Concept Bottleneck Models (CBMs) (Koh et al., 2020) established this paradigm, and Concept Embedding Models (CEMs) (Zarlenga et al., 2022) extended it by encoding concepts as learned embeddings. Both CBMs and CEMs have seen a rising uptake in domains ranging from medical imaging (Wang et al., 2024; Chanda et al., 2024; Patrício et al., 2025; Bunnell et al., 2024; Qu et al., 2026; Monnin et al., 2026; Wang et al., 2026) to autonomous driving (Echterhoff et al., 2023) where interpretability is critical.

Interventions may fail when concept representations carry **Information Leakage**: content beyond the concept's semantic definition (Kazhdan et al., 2021; Margeloiu et al., 2021; Mahinpei et al., 2021; Havasi et al., 2022; Parisini et al., 2026). A model tasked with assessing prognosis in invasive skin cancers may leak information about tumour size into a proliferation rate concept representation, because the two are correlated in training data. A clinician intervening on proliferation rate — to ask what would happen if this tumour were less aggressive — would receive a misleading answer since the concept representation cannot be interpreted independently of size. More fundamentally, in the presence of leakage, the clinician's correction may have little effect on model performance and in some cases can negatively impact task performance (Zarlenga et al., 2024).

Existing approaches tackle information leakage through training the concept encoder and task head independently or sequentially, or through introducing some additional architectural components that increase model complexity. In this work we show that a principal leakage mechanism is a consequence of the optimisation objective used in training concept-based models. During joint training,

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.

the concept encoder receives a task-driven gradient at every training step, pushing it to encode spurious task-relevant features into concept representations.

We propose **Gradient-Cancelled (GC) Concept-Based Models**, a general training framework that cancels this gradient term for encoder parameters throughout training resulting in an encoder trained on concept supervision alone. GC trades a small reduction in pre-intervention task accuracy (which we attribute to leakage) for substantially improved post-intervention performance and consistently higher transparency. This trade-off is particularly favourable in high-risk domains where expert correction is the operating paradigm. Our method is a minimal modification to the joint training objective and can extend to any concept-based model with a concept encoder that produces a concept representation and passes it to a task head.

Our contributions:

- **Gradient Cancellation: a general training framework.** We introduce GC which removes the task-driven gradient from concept encoder updates via a Gradient Reversal Layer (GRL). GC requires no new major architectural components or optimisation steps and applies to any concept-based model with a concept representation passed from an encoder to a task head. To demonstrate the impact of our framework, we focus on two representative instantiations: GC-CBM which provably recovers the sequential CBM optima within a single joint training run; and GC-CEM, the first training framework to reduce leakage in CEMs without architectural modification. Across real world datasets (CUB and AwA2), GC-trained models show significantly stronger response to human correction when compared to the vanilla model variants. We further outline how GC can be applied to other commonly used CM architectures.
- **Mechanistic explanation of leakage.** Previous work has identified conditions that increase leakage, including incomplete concept sets (Havasi et al., 2022) and soft concept representations (Mahinpei et al., 2021). We introduce an information-theoretic framework showing that, under task-head convergence, joint training induces an explicit mutual-information maximisation term in the encoder update. This also provides a theoretical connection between the optimisation dynamics and concept incompleteness, showing how incomplete concept sets admit a larger task-driven leakage signal.
- **Embedding-level leakage metrics RTL and RCL**. Given the difficulty of direct mutual information estimation at high dimensionality, we introduce two new functional metrics, Residual Task Leakage (RTL) and Residual Interconcept Leakage (RCL), to measure concept-task and interconcept leakage in CMs with a vector concept representation. Our metrics can also be applied in supervised representation learning and for dictionary learning approaches commonly used in mechanistic interpretability.

2 BACKGROUND

Concept-based Models. Concept-based models produce a task prediction and a set of concept scores for each input. Many CMs can be expressed as a generalised Concept Bottleneck Model (Koh et al., 2020): a triple (g_θ, s, f_ϕ) consisting of a concept encoder $g : \mathbb{R}^n \to \mathbb{R}^{k\times d}$ parameterised by θ producing k concept representations $\hat{c} = (\hat{c}_1, \ldots, \hat{c}_k)$ with $\hat{c}_i \in \mathbb{R}^d$; a scoring function $s : \mathbb{R}^{k\times d} \to [0,1]^k$ mapping representations to concept scores $\hat{p} = s(\hat{c})$ and a task head $f : \mathbb{R}^{k\times d} \to [0,1]^L$ parameterised by ϕ, predicting $\hat{y} = f(\hat{c})$.

CMs are trained using a dataset $\mathcal{D} = \{(x^{(n)}, c^{(n)}, y^{(n)})\}_{n=1}^N$ consisting of an input, a task label and k binary concept annotations $c^{(n)} \in \{0,1\}^k$. Training minimises a task loss $\mathcal{L}_y(f(g(x)), y)$ and a concept loss $\mathcal{L}_c(s(g(x)), c)$, optimised independently, sequentially (both defined in Appendix A) or jointly. Joint training, typically attains the strongest task performance but is most prone to leakage:

$$\mathcal{L}_{\text{CM-joint}}(\theta, \phi) = \mathcal{L}_y(f_\phi(g_\theta(x)), y) + \lambda_c \mathcal{L}_c(s(g_\theta(x)), c). \tag{1}$$

Architectures differ by choice of (d, s); we focus on the two most widely used.

Sigmoidal and Hard CBMs The Sigmoidal CBM (Koh et al., 2020) sets $d = 1$ and $s = \mathrm{id}$ so that each concept is a single sigmoidal scalar and the bottleneck is the score vector ($\hat{c} = \hat{p}$). Concept supervision directly acts on the bottleneck vector and so concepts are maximally constrained in

expressiveness. Hard CBMs by construction cannot encode any leaked information: they are trained independently, and concept predictions are binarised before being fed into the task head.

CEMs. Concept Embedding Models (Zarlenga et al., 2022) set $d > 1$ with a learned scoring function that can be shared or concept specific. Each concept i has prototype generators $\rho_i^+, \rho_i^- \in \mathbb{R}^{d'}$ which are linearly combined to produce a concept representation $\hat{c}_i = \hat{p}_i \rho_i^+ + (1 - \hat{p}_i)\rho_i^-$. For CEMs the scoring function associated with c_i receives the concatenated concept prototypes such that $\hat{p}_i = s_i(\rho_i) = s_i([\rho_i^+, \rho_i^-])$ and $d = 2d'$. CEMs were designed with richer concept representations to recover predictive performance lost by enabling leakage (Zarlenga, 2026), whilst retaining concept probabilities $\hat{p}_i$ for human supervision and interventions. Several extensions of CEMs have been proposed (Zarlenga et al., 2024; Espinosa Zarlenga et al., 2025; Hill et al., 2026); our GC framework applies to each (Appendix B). To our knowledge, no prior work on CEMs has proposed a training framework which reduces leakage whilst significantly improving responsiveness to interventions without major architectural modification.

Concept Interventions. A central property of CMs is test-time concept interventions. An intervention replaces predicted $\hat{c}_i$ with the ground truth label c_i and propagates this corrected representation to the label predictor. There is also a growing body of literature focusing on intervention policies indicating the order of the concepts to intervene on (Chauhan et al., 2023; Shin et al., 2023).

Information Leakage. The phenomenon of information leakage was first identified by Mahinpei et al. (2021), who show that soft concept representations can carry information beyond the intended concepts. Parisini et al. (2026) provides a taxonomy of leakage into Concept-Task Leakage and Interconcept Leakage, alongside mutual information based estimators for both. Zarlenga (2026) argue for a notion of 'benign leakage', where additional task-relevant information encoded into concept representations may be used to improve intervention performance.

There is a growing body of research focusing on alleviating leakage in concept-based models. Sun et al. (2024) introduce SupCBM, a new paradigm applicable to CBMs that mitigates leakage by incorporating label supervision into concept prediction and constructing a hierarchical concept set. Galliamov et al. (2026) and Almudévar et al. (2026) take information-theoretic approaches to reducing leakage by introducing Information Bottleneck regularisers to restrain concept representations. Moreau et al. (2026) use kernels to approximate mutual information estimates of interconcept leakage and minimise it through an additional loss term. In contrast, our approach reduces leakage through a single training-time gradient modification, with no additional loss terms or components.

3 Gradient-Cancelled Concept-Based Models

Joint training encourages the concept encoder to predict annotated concepts while retaining information useful for the task, potentially extending beyond the intended concepts. We first characterise this pressure through an information-theoretic analysis under an optimal task head. We then introduce the GC framework, which removes the task-loss contribution to the concept encoder updates at every training step, without requiring task-head convergence. This trains the encoder solely through concept supervision while fitting the task head concurrently to its evolving representations, and generalises to any CM of the form (g, s, f). We establish its relationship to sequential training and evaluate how removing this source of pressure affects leakage, predictive performance, and interventions.

3.1 Leakage in Joint Training

We follow the leakage taxonomy and measures of Parisini et al. (2026), who classify leakage into two types: **Concepts-Task Leakage** (CTL), where additional information about the task is stored in the learned concepts; and **Interconcept Leakage** (ICL), where additional information about the other concepts is stored into each learned concept, resulting in learned concepts being more predictive of the value of another learned concept.

Gradient Analysis. We now show that information leakage is encouraged by joint training since the concept encoder is explicitly incentivised to maximise mutual information between concept

representations and task labels. In Appendix C we also provide a theoretical link between this mechanism and how it is affected by concept incompleteness, a well known condition that increases leakage (Kazhdan et al., 2021; Margeloiu et al., 2021; Mahinpei et al., 2021; Havasi et al., 2022; Parisini et al., 2026).

Assuming the task loss is cross-entropy loss, the task head is trained to minimise the expected cross entropy over the data distribution:

$$\mathbb{E}_{(\hat{c}\sim\hat{C},y\sim Y)}[-\log f_\phi(y \mid \hat{c})] = H(Y \mid \hat{C}) + \mathbb{E}_{\hat{c}\sim\hat{C}}\left[\mathrm{KL}\big(p(\cdot \mid \hat{c}) \,\big\|\, f_\phi(\cdot \mid \hat{c})\big)\right].$$

As the second term is minimised the task head converges to the Bayes predictor $f_{\phi^*}(\cdot \mid \hat{c}) = p(\cdot \mid \hat{c})$. Using the identity $H(Y|\hat{C}) = H(Y) - I(\hat{C};Y)$ and taking the gradient with respect to encoder parameters θ, we obtain the following:

$$\nabla_\theta \mathcal{L}_{\text{CM-joint}} = -\nabla_\theta I(\hat{C};Y) + \lambda_c \nabla_\theta \mathcal{L}_c(s(\hat{c}), c). \tag{2}$$

Assuming task head convergence, the concept encoder parameters are updated to implicitly maximise $I(\hat{C};Y)$ through encoding any feature, concept-aligned or spurious, that increases final task performance. Our GC framework cancels this term out throughout training so that encoder parameters are trained solely from concept supervision.

The Mutual Information (MI) maximisation term in equation 2 directly encourages $I(\hat{C};Y)$ to increase beyond the desired ground truth value $I(C;Y)$ resulting in non-zero CTL. ICL is understood as largely a side effect of CTL (Parisini et al., 2026) and so removing this MI term indirectly reduces ICL.

3.2 Gradient Cancellation Training

A natural first approach introduces an explicit adversary h_ψ, identical in architecture to the task head i.e., $\{f_\phi\} = \{h_\psi\}$, trained to cancel the task gradient at the bottleneck (Appendix D). Cancellation can only occur if h_ψ tracks f_ϕ exactly, and any mismatch during training results in leakage. We therefore enforce $\{f_\phi\} = \{h_\psi\}$ by construction: the adversarial branch is the task head itself, with its gradient reversed at the concept bottleneck via a GRL.

GRL. The GRL (Ganin & Lempitsky, 2015) is a 'pseudo-function' R_λ that acts as an identity function during the forward pass, and scales the incoming gradient of the loss with respect to its input by $-\lambda$ in the backward pass.

In our training paradigm we utilise a GRL as a map $R_\lambda : \mathbb{R}^{k\times d} \to \mathbb{R}^{k\times d}$ on the codomain of a given CMs concept encoder. The concept representations $\hat{c}$ reach the task head along two parallel paths: directly, as $f_\phi(\hat{c})$, and through the GRL as $f_\phi(R_\lambda(\hat{c}))$. During the backward pass we obtain the following:

$$\nabla_\theta \, \mathcal{L}_y\big(f_\phi(R_\lambda(g_\theta(x))), y\big) = -\lambda \, \nabla_\theta \, \mathcal{L}_y\big(f_\phi(g_\theta(x)), y\big),$$

which recovers the adversary's contribution and gives the following training objective:

$$\mathcal{L}_{\text{GC-CM}} = \mathcal{L}_y(f_\phi(g_\theta(x)), y) + \lambda_c \mathcal{L}_c(s(g_\theta(x)), c) + \mathcal{L}_y(f_\phi(R_\lambda(g_\theta(x))), y).^1 \tag{3}$$

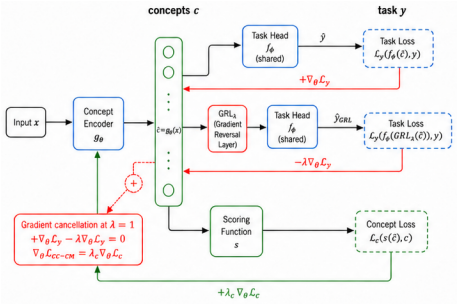


Figure 1: GC-CM Architecture

[1]Both task branches share the task head, resulting in a doubled gradient update which we account for by rescaling the learning rate; Proposition 1 uses the averaged form of equation 3 for exact equivalence.

Given we have exact cancellation at $\lambda = 1$, we arrive at the following concept encoder update at every step rather than only when f_ϕ and h_ψ converge:

$$\nabla_\theta \mathcal{L}_{\text{GC-CM}} = \lambda_c \nabla_\theta \mathcal{L}_c(s(g_\theta(x), c). \tag{4}$$

The concept encoder now learns from concept supervision alone and the task head update is unaffected if both task loss terms are scaled by $\frac{1}{2}$, performing cross entropy minimisation of predicted labels $\hat{y}$. We provide a full derivation showing GC training removes this task-based gradient in CEMs at both the vector and the probability level in Appendix E.2.

λ controls the strength of task supervision reaching the concept encoder, interpolating between joint training ($\lambda = 0$) and full cancellation ($\lambda = 1$). At full cancellation, the encoder updates are equivalent to those obtained with a detached bottleneck. Intermediate values retain a task-driven learning signal, allowing the encoder to balance concept fidelity with task performance. This flexibility may be useful in setups where some leakage is acceptable and beneficial, as discussed by Zarlenga (2026). We examine how leakage varies with λ in a controlled Gaussian experiment in Appendix F.2.

3.3 Theoretical Properties

In this subsection we present the key theoretical properties of the GC framework. We provide proofs to all propositions in Appendix G.

Proposition 1 (GC-CBM Optima Equivalence with Sequential CBM)**.** *Set $d = 1$, $s = \text{id}$ and let $\{\theta_t\}_{t=1}^T$ be the sequence of concept encoder parameters under GC-CBM training. Under the GC training framework with $\lambda = 1$, the concept encoder gradient update direction is equivalent to the Sequential CBM concept encoder update direction at all training steps:*

$$\nabla_\theta \mathcal{L}_{GC-CBM} = \lambda_c \nabla_\theta \mathcal{L}_c(g_\theta(x), c) = \lambda_c \nabla_\theta \mathcal{L}_{Seq}.$$

Assuming the concept encoders convergence to the optimal parameters $\lim_{t\to\infty} \theta_t = \theta^*$, and $\hat{c} \mapsto \nabla_\phi \mathcal{L}_y(f_\phi(\hat{c}), y)$ is Lipschitz, the task head converges asymptotically to the sequential CBM task head optimum:

$$\lim_{t\to\infty} \phi_t^{GC} = \phi_{seq}^* = \arg\min_\phi \mathbb{E}[\mathcal{L}_y(f_\phi(g_{\theta^*}(x)), y)].$$

GC-CBM converges to the same concept encoder and task head stationary points as the Sequential CBM without the need for a two-stage training procedure.

Although the two procedures share the same optima, they attain them through different paths: the GC task head is fit across the encoder's trajectory, seeing the sequence of bottleneck distributions $\{p(\hat{c} \mid \theta_t)\}_{t=1}^T$, whereas the sequential head is fit once to a frozen encoder $g_{\theta^*}(x)$. We hypothesise that exposure to evolving concept representations makes the GC task head more robust to test-time interventions. Our empirical results provide support for this hypothesis, showing improved intervention performance when compared with sequential training.

Proposition 2 (λ_c Invariance)**.** *Under the GC framework with $\lambda = 1$, the encoder update direction is independent of $\lambda_c > 0$:*

$$\nabla_\theta \mathcal{L}_{GC\text{-}CM} = \lambda_c \nabla_\theta \mathcal{L}_c.$$

The weight λ_c acts as a scalar multiplier on the concept gradient - equivalent to scaling the concept encoder learning rate by λ_c. The concept encoder therefore converges to the same optima θ^* for all $\lambda_c > 0$ provided the effective learning rate $\eta_\theta \lambda_c$ satisfies standard learning rate stability conditions. The task head update (just like all other training methods) is independent of λ_c.

This learning-rate interpretation connects to a limiting regime of joint CBM training, which we discuss in Appendix G.2. In vanilla CBM and CEM training λ_c must be carefully balanced against the task objective to prevent concepts being ignored or task performance degrading. Our training setup is robust to the choice of λ_c. We validate this empirically in Section 5.

4 Evaluation Metrics

We use an evaluation suite based on four main CM desiderata:

Task performance. A CM must maintain strong task performance to be useful. We measure task performance through test accuracy on the task label y.

Concept Fidelity. Learned concepts must faithfully reflect their defined semantic meaning. This is measured by concept prediction accuracy.

Intervenability. A central promise of concept-based models is the ability for human experts to correct incorrect concept predictions at test time and therefore improve downstream task predictions (Banerji et al., 2025; Zarlenga, 2026). We evaluate intervenability by measuring the gain in task accuracy when wrong predicted concepts are corrected to their ground truth values at test time. A model that relies on faithful concept representations should show large intervention gains whereas a model with high leakage should show smaller gains since the task head relies on hidden information not accounted for by ground-truth concepts (Parisini et al., 2026).

Leakage. Learned concepts must not encode additional information beyond their intended semantic meaning. We evaluate for leakage at the concept probability level and also at the concept vector level for embedding based models. At the probability level we use the CTL and ICL scores introduced in (Parisini et al., 2026) (see Appendix H). We compute these via Kraskov–Stögbauer–Grassberger (KSG) mutual information estimators (Kraskov et al., 2004). At higher dimensions, MI and entropy estimators are heavily biased due to the curse of dimensionality (Lord et al., 2018; Carrara & Ernst, 2019; Czyż et al., 2023; Gowri et al., 2024). Thus, we introduce new functional measures akin to CTL and ICL to check for leakage in concept vectors:

Residual Task Leakage (RTL) *and Residual Interconcept Leakage* (RCL). We introduce functional metrics that detect concept-task and interconcept leakage at the embedding level. We standardise embeddings using training-set statistics and the following two steps: (1) for concept i, we regress out the contribution of c_i from the embedding via Ridge regression and obtain the following residuals $\mathbf{R}_i = \hat{c}_i - \mu(c_i)$. (2) We then fit a one-hidden-layer MLP predicting $\mathbf{R}_i$ from either the task label y (RTL) or another concept c_j (RCL), and compute the per-dimension coefficient of determination R^2_m.

Residual Task Leakage (RTL) measures excess task-predictive information in concept vector $\hat{c}_i$ beyond what the ground truth concept c_i explains. For concept vector $\hat{c}_i$, we regress out the contribution of c_i from the embedding via Ridge regression and obtain the following residuals $\mathbf{R}_i = \hat{c}_i - \mu(c_i)$. We then use a small one-hidden-layer MLP to fit task label y to residual $\mathbf{R}_i$ and compute a per dimension coefficient of determination R^2_m. We weight each dimension by its residual variance $\sigma^2_m = \mathrm{Var}(\mathbf{R}_{i,m})$ and clamp at zero to obtain the following:

$$\mathrm{RTL}_i = \frac{1}{d}\sum_{m=1}^{d} \max\left(0,\ R^2_m(y \to \mathbf{R}_i)\right)\ \sigma^2_m, \qquad \mathrm{RTL} = \frac{1}{k}\sum_{i=1}^{k} \mathrm{RTL}_i. \tag{5}$$

$\mathrm{RTL}_i > 0$ implies the residual information in the learned concept vector can be explained by the task labels and is therefore task-predictive.

Residual Interconcept Leakage (RCL) measures excess interconcept predictiveness between concept vectors. Given a concept pair (i, j) we first regress out the contribution of c_i from the embedding $\hat{c}_i$ and obtain residuals $\mathbf{R}_i = \hat{c}_i - \mu(c_i)$. We then use a one-hidden-layer MLP to fit the ground-truth concept c_j onto this residual $\mathbf{R}_i$ and use this to compute a per-dimension coefficient of determination R^2_m weighted by residual variance $\sigma^2_m = \mathrm{Var}(\mathbf{R}_{i,m})$ and clamped at zero. We also normalise by embedding dimension d to obtain:

$$\mathrm{RCL}_{ij} = \frac{1}{d}\sum_{m=1}^{d} \max\left(0,\ R^2_m(c_j \to \mathbf{R}_i)\right)\ \sigma^2_m, \qquad \mathrm{RCL} = \frac{1}{k(k-1)}\sum_{i}\sum_{j\neq i} \mathrm{RCL}_{ij}. \tag{6}$$

We measure how well the ground truth concept c_j explains the variance of the residual embedding for concept i. A positive RCL_{ij} indicates that the ground truth concept c_j explains the structure in residual embedding $\mathbf{R}_i$ beyond what c_i itself accounts for. RCL therefore provides a proxy for measuring additional interconcept information at the embedding level. Ridge regression is sufficient for the first regression because, for binary c_i, the optimal predictor is the class-conditional mean which is linear. In the second step we use shallow MLPs to ensure we detect non-linear leakage. All models are fit to the training set and evaluated on the test set. Validation of both metrics on controlled Gaussian datasets is provided in Appendix I.

5 Results and Discussion

In this section we present our experimental results for our training framework specifically for sigmoidal CBMs and CEMs. We evaluate all models on TabularToy, AwA2, CUB-200 across three values of $\lambda_c \in \{0.1, 0.5, 1\}$; full details of datasets, architectures, training procedures and hyperparameters are provided on Appendix J. We start by evaluating GC-CBM and compare our training setup to joint CBM training with Hard, Sequential as architectural references. Hard CBM is trained independently and concept probabilities are binarised at test time before being passed to the task head.

5.1 GC-CBM Results

GC-CBM achieves strong concept fidelity across all settings. Figure 2 shows that GC-CBM produces highly faithful concepts with higher concept accuracy across all λ_c when compared to joint training for CUB and TabularToy. Concept accuracy is competitive with joint and sequential CBM across all datasets and in some cases outperforms them (for AwA2 and CUB against both references for all λ_c). Moreover, our results complement Proposition 2, which theoretically shows concept fidelity is invariant to λ_c for GC-CBM if learning rate stability conditions are satisfied. Our model therefore removes the need for λ_c hyperparameter tuning.

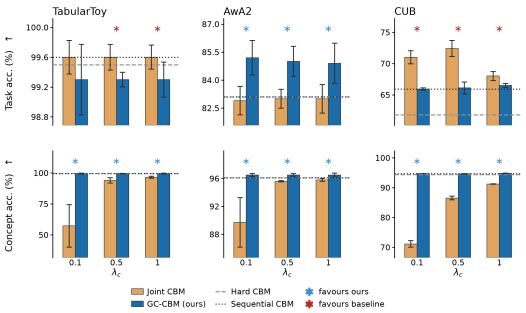


Figure 2: GC-CBM task and concept accuracy vs. Joint CBM. Asterisks mark comparisons between GC-CBM and Joint CBM

GC-CBM reduces leakage whilst maintaining competitive task accuracy. GC-CBM lowers CTL and ICL substantially against joint training across datasets (Figure 3), with CTL scores lower than Hard and Sequential CBMs on CUB. At the same time, task performance remains strong (Figures 2). GC training yields significant task performance gains on AwA2 over all λ_c (+2.07% average), while on CUB we observe a task performance drop (70.5% vs 66.2%), which we attribute to the removal of spurious task-relevant features and to improved model transparency. This is consistent with our theoretical expectations: the largest reduction in CTL is accompanied by the largest drop in task accuracy, while the smallest reduction incurs little or no accuracy loss.

Gradient cancellation makes concept interventions more effective. GC-CBM benefits from concept interventions across all three datasets, whereas jointly trained CBMs can suffer substantial accuracy losses, particularly under weak concept supervision. Under full intervention on CUB, GC-CBM also outperforms sequential training (98.07% vs. 96.59%), consistent with our hypothesis that training the task head on evolving concept representations improves intervention robustness. On AwA2, GC-CBM achieves 100% post-intervention accuracy with zero variance, compared with $85.96 \pm 9.88\%$ for joint training, despite a modest reduction in measured CTL.

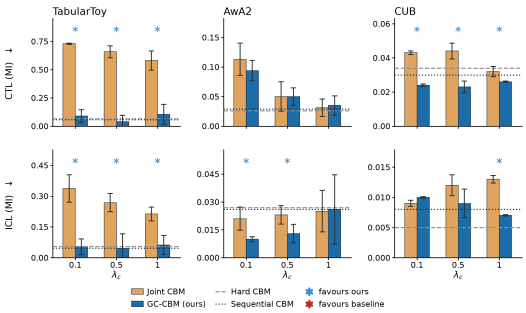


Figure 3: GC-CBM concepts-task and interconcept leakage vs Joint CBM. Asterisks mark comparisons between GC-CBM and Joint CBM

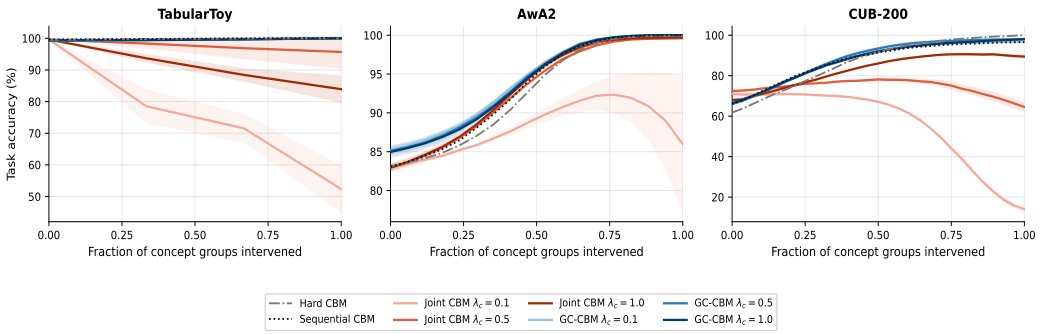


Figure 4: Plot of task performance against concept interventions for GC-CBM, Hard, Sequential and Joint CBMs on TabularToy, AwA2 and CUB.

5.2 GC-CEM RESULTS

We now evaluate our second instantiation, GC-CEM, against vanilla CEM training.

GC-CEM improves concept fidelity while preserving competitive task performance. GC trained models achieve improvements in concept accuracy across both real world-datasets across all λ_c (significant at all λ_c for CUB and at $\lambda_c \in \{0.5, 1\}$ for AwA2). Task performance remains competitive with CEMs with accuracy drops at most of 3.8% (CUB), while outperforming CEM task performance on AwA2 at $\lambda_c \in \{0.5, 1\}$.

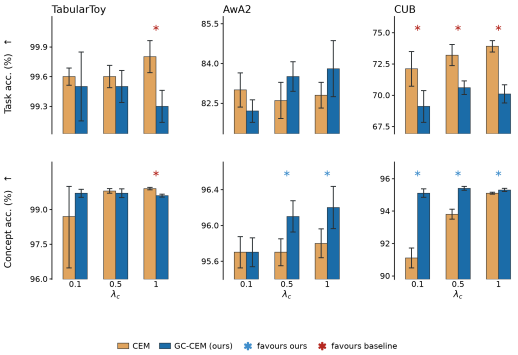


Figure 5: GC-CEM task and concept accuracy vs CEM. Asterisks mark comparisons between GC-CEM and Joint CEM.

Gradient cancellation suppresses leakage in concept embeddings. GC training substantially reduces RTL relative to vanilla CEM across all datasets and λ_c settings: by 98–99% on TabularToy, 91–98% on AwA2, and 40–51% on CUB (Figure 6). RCL decreases by 97–98% on TabularToy and 91–99% on AwA2, but increases slightly on CUB.

GC-CEM improves intervention performance. Similar to GC-CBM, GC-CEM shows stronger task improvements with increasing interventions on CUB and AwA2 when compared to vanilla CEM. On CUB, GC-CEM attains a post-intervention accuracy of up to **87.4%**, significantly higher than any vanilla CEM configuration, with intervention gains of **17.3%** versus **7.2%** for vanilla CEM. On AwA2, GC-CEM improves under intervention across all λ_c (+1.4 to +2.3 pp) while vanilla CEM is essentially flat (-0.1 to +0.6 pp). Although all models sit at the ceiling task accuracy for TabularToy, interventions on vanilla CEM result in degraded task performance (-0.1%) while GC-CEM improves (+0.4%).

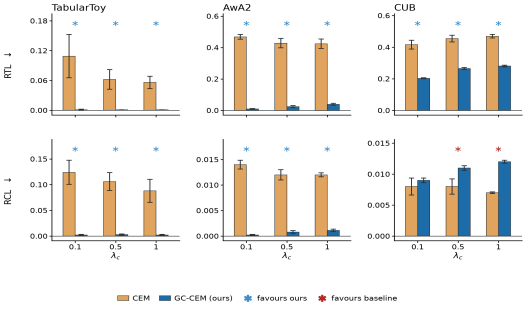


Figure 6: GC-CEM RTL and RCL vs CEM across all datasets. Asterisks mark comparisons between GC-CEM and Joint CEM.

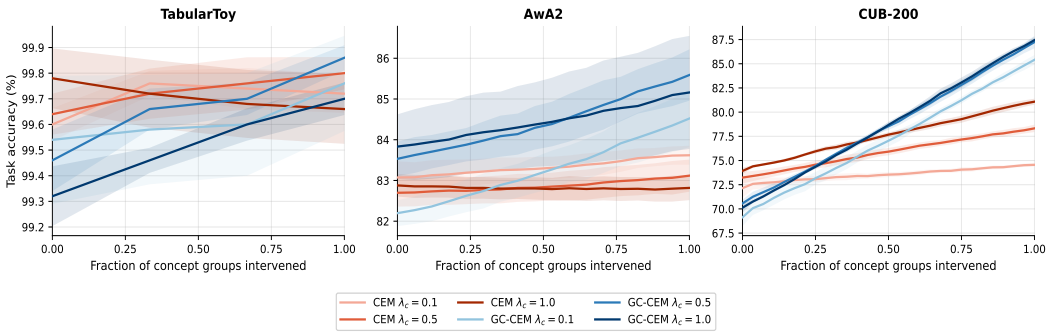


Figure 7: Task accuracy as a function of the number of concept interventions for CEM and GC-CEM on TabularToy, AwA2 and CUB.

These results demonstrate that removing task-driven leakage produces more faithful concept representations that supports stronger human-in-the-loop correction.

6 Conclusion and Limitations

In this work we introduce **Gradient Cancelled Concept-Based Models**, and study in detail two instantiations of this general training framework (GC-CBM and GC-CEM). We develop an information-theoretic characterisation of leakage in jointly trained concept models, showing that, under task-head convergence, the task-driven encoder update corresponds to an explicit mutual-information maximisation term and relating this mechanism to concept incompleteness. GC removes this task-driven signal by introducing a GRL as an additional path to the task head, leaving the concept encoder to learn from concept supervision alone without additional training stages or core architectural modification.

We show that GC-CBM encoder follows the sequential CBM update direction at each step and the task head converges asymptotically to the sequential optimum, recovering both within a joint training framework. GC-CEM provides the first training setup for CEMs that substantially reduces leakage in concept representations without architectural modification. Across both instantiations we improve intervention performance and concept fidelity whilst maintaining strong task performance. Our GC framework generally applies to any end-to-end trained concept-based model where task gradients flow through a concept representation to a shared concept encoder, including IntCEMs (Zarlenga et al., 2024), Hierarchical CEMs (Hill et al., 2026), Variational CEMs (De Santis et al., 2026), CBM-AUC (Sawada & Nakamura, 2022) and Stochastic CBMs (Vandenhirtz et al., 2024) amongst others. We outline the GRL placement for each of these architectures in Appendix B.

We also introduce RTL and RCL, two functional metrics for quantifying leakage in high-dimensional concept embeddings, where mutual information estimation is unreliable. By measuring whether learned representations contain structure beyond that explained by their supervised labels, RTL and RCL provide a principled way to assess embedding leakage in CEMs and, more broadly, in supervised representation learning (Song & Shmatikov, 2020). Our metrics may also be applied in mechanistic interpretability, where features recovered by dictionary learning are assigned concept labels (Bricken et al., 2023); RCL may further offer a supervised test of whether such a feature is monosemantic or encodes other concept information, as expected under superposition (Elhage et al., 2022).

Limitations. Gradient cancellation removes the task-driven leakage mechanism however it does not eliminate residual leakage arising from soft probabilities, which may encode distributional information about the input. This residual leakage is secondary to the source we focus on in this work and exists across all CBM training paradigms and only depends on the choice of concept representations.

AI USE STATEMENT

In this work, we used generative AI tools to implement methods, generate synthetic datasets, provide critical ingredients in proving mathematical claims. We have not used generative AI tools to help develop theoretical models or conceptual frameworks, formulate mathematical claims, assist in the writing of proofs, propose or refine hypotheses, design or provide feedback on research methodology or experiments, clean and reformat dataset, support qualitative and thematic data analysis, interpret results, and assisting with translation is not applicable to this work. Additionally, we used generative AI tools to create or modify scientific figures or images, create or edit software code, creation of artifacts and to edit a research paper to improve readability. We have reviewed all AI-assisted work. We ensure all ingredients towards mathematical claims are manually reviewed and where possible align with existing literature. We use AI tools to generate the toy dataset used in Appendix F.2 and review the generated code to ensure it follows the steps outlined in the Appendix. All AI generated code is reviewed manually before being pushed. Only Figure 1 was AI generated as a schematic and was reviewed manually to ensure it matched requirements. We use AI tools to edit LaTeX code for better readability and review each edit made manually. AI generated artifacts include training and evaluation code and all changes have been manually reviewed. 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 training and architectural details for all experiments in Appendix J including learning rates, batchsize, architectures, embedding dimensions, optimiser and epoch counts. We outline the concept grouping we use for interventions on AwA2 in Appendix J.1. A link to an anonymous downloadable source code with experiment configurations is provided in supplementary material.

REFERENCES

Antonio Almudévar, José Miguel Hernández Lobato, and Alfonso Ortega. There was never a bottleneck in concept bottleneck models. In *International Conference on Learning Representations*, volume 2026, pp. 114146–114165, 2026.

Christopher R. S. Banerji, Tapabrata Chakraborti, Aya Abdelsalam Ismail, Florian Ostmann, and Ben D. MacArthur. Train clinical ai to reason like a team of doctors. *Nature*, 639:32–34, 2025. doi: 10.1038/d41586-025-00618-x.

Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, et al. Towards monosemanticity: Decomposing language models with dictionary learning. Transformer Circuits Thread, 2023. URL `Https://transformer-circuits.pub/2023/monosemantic-features/index.html`.

Arianna Bunnell, Yannik Glaser, Dustin Valdez, Thomas K. Wolfgruber, Aleen Altamirano, Carol Zamora González, Brenda Y. Hernandez, Peter Sadowski, and John A. Shepherd. Learning a clinically-relevant concept bottleneck for lesion detection in breast ultrasound. In *Medical Image Computing and Computer Assisted Intervention – MICCAI 2024*, volume 15003 of *Lecture Notes in Computer Science*, pp. 650–659. Springer Nature Switzerland, 2024. doi: 10.1007/978-3-031-72384-1_61.

Nicholas Carrara and Jesse Ernst. On the estimation of mutual information, 2019. URL `https://arxiv.org/abs/1910.00365`.

Tirtha Chanda, Katja Hauser, Sarah Hobelsberger, Tabea-Clara Bucher, Carina Nogueira Garcia, Christoph Wies, Harald Kittler, Philipp Tschandl, Cristian Navarrete-Dechent, Sebastian Podlipnik, Emmanouil Chousakos, Iva Crnaric, Jovana Majstorovic, Linda Alhajwan, Tanya Foreman, Sandra Peternel, Sergei Sarap, İrem Özdemir, Raymond L. Barnhill, Mar Llamas-Velasco, Gabriela Poch, Sören Korsing, Wiebke Sondermann, Frank Friedrich Gellrich, Markus V. Heppt, Michael Erdmann, Sebastian Haferkamp, Konstantin Drexler, Matthias Goebeler, Bastian Schilling, Jochen S. Utikal, Kamran Ghoreschi, Stefan Fröhling, Eva Krieghoff-Henning, Reader Study Consortium, and Titus J. Brinker. Dermatologist-like explainable ai enhances

trust and confidence in diagnosing melanoma. *Nature Communications*, 15:524, 2024. doi: 10.1038/s41467-023-43095-4.

Kushal Chauhan, Rishabh Tiwari, Jan Freyberg, Pradeep Shenoy, and Krishnamurthy Dvijotham. Interactive concept bottleneck models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 5948–5955, 2023. doi: 10.1609/aaai.v37i5.25736.

Paweł Czyż, Frederic Grabowski, Julia E. Vogt, Niko Beerenwinkel, and Alexander Marx. Beyond normal: On the evaluation of mutual information estimators, 2023. URL https://arxiv.org/abs/2306.11078.

Francesco De Santis, Gabriele Ciravegna, Philippe Bich, Danilo Giordano, and Tania Cerquitelli. V-CEM: Bridging performance and intervenability in concept-based models. In *Explainable Artificial Intelligence*, volume 2576 of *Communications in Computer and Information Science*, pp. 48–67. Springer, 2026. doi: 10.1007/978-3-032-08317-3_3. URL https://doi.org/10.1007/978-3-032-08317-3_3.

Jessica Echterhoff, An Yan, Kyungtae Han, Amr Abdelraouf, Rohit Gupta, and Julian McAuley. Driving through the concept gridlock: Unraveling explainability bottlenecks in automated driving, 2023. URL https://arxiv.org/abs/2310.16639.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, Roger Grosse, Sam McCandlish, Jared Kaplan, Dario Amodei, Martin Wattenberg, and Christopher Olah. Toy models of superposition, 2022. URL https://arxiv.org/abs/2209.10652.

Mateo Espinosa Zarlenga, Gabriele Dominici, Pietro Barbiero, Zohreh Shams, and Mateja Jamnik. Avoiding leakage poisoning: Concept interventions under distribution shifts. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 15564–15595. PMLR, 2025.

Karim Galliamov, Syed M Ahsan Kazmi, Adil Khan, and Adín Ramírez Rivera. Concepts' information bottleneck models, 2026. URL https://arxiv.org/abs/2602.14626.

Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation, 2015. URL https://arxiv.org/abs/1409.7495.

Gokul Gowri, Xiao-Kang Lun, Allon M. Klein, and Peng Yin. Approximating mutual information of high-dimensional variables using learned representations, 2024. URL https://arxiv.org/abs/2409.02732.

Marton Havasi, Sonali Parbhoo, and Finale Doshi-Velez. Addressing leakage in concept bottleneck models. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=tglniD_fn9.

Oscar Hill, Mateo Espinosa Zarlenga, and Mateja Jamnik. Hierarchical concept-based interpretable models, 2026. URL https://arxiv.org/abs/2602.23947.

Dmitry Kazhdan, B. Dimanov, Helena Andrés-Terré, Mateja Jamnik, Pietro Lio', and Adrian Weller. Is disentanglement all you need? comparing concept-based & disentanglement approaches. *CoRR*, abs/2104.06917, 2021.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. URL https://arxiv.org/abs/1412.6980.

Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. Concept bottleneck models, 2020. URL https://arxiv.org/abs/2007.04612.

Alexander Kraskov, Harald Stögbauer, and Peter Grassberger. Estimating mutual information. *Phys. Rev. E*, 69:066138, Jun 2004. doi: 10.1103/PhysRevE.69.066138. URL https://link.aps.org/doi/10.1103/PhysRevE.69.066138.

Warren M. Lord, Jie Sun, and Erik M. Bollt. Geometric k-nearest neighbor estimation of entropy and mutual information. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 28(3), March 2018. ISSN 1089-7682. doi: 10.1063/1.5011683. URL http://dx.doi.org/10.1063/1.5011683.

Anita Mahinpei, Justin Clark, Isaac Lage, Finale Doshi-Velez, and Weiwei Pan. Promises and pitfalls of black-box concept learning models, 2021. URL https://arxiv.org/abs/2106.13314.

Andrei Margeloiu, Matthew Ashman, Umang Bhatt, Yanzhi Chen, Mateja Jamnik, and Adrian Weller. Do concept bottleneck models learn as intended?, 2021. URL https://arxiv.org/abs/2105.04289.

Killian Monnin, Patrick Jeltsch, Lucia Fernandes-Mendes, Vasco Cazzagon, Murat Yüce, Vivek Yadav, Mario Jreige, Marianna Gulizia, Montserrat Fraga Christinet, Raphaël Girardet, Clarisse Dromain, Bachir Taouli, Naïk Vietti-Violi, and Jonas Richiardi. Explainable hcc diagnosis on dynamic contrast-enhanced mri with a li-rads concept bottleneck. In *Proceedings of The 9th International Conference on Medical Imaging with Deep Learning*, volume 315 of *Proceedings of Machine Learning Research*, pp. 3283–3313. PMLR, 2026.

Pierre Moreau, Emeline Pineau Ferrand, Yann Choho, Benjamin Wong, Annabelle Blangero, and Milan Bhan. Towards faithful multimodal concept bottleneck models, 2026. URL https://arxiv.org/abs/2603.13163.

Enrico Parisini, Tapabrata Chakraborti, Chris Harbron, Ben D. MacArthur, and Christopher R.S. Banerji. Leakage and interpretability in concept-based models. *Journal of Machine Learning Research*, 27(174):1–40, 2026. URL http://jmlr.org/papers/v27/25-1121.html.

Cristiano Patrício, Luís F. Teixeira, and João C. Neves. A two-step concept-based approach for enhanced interpretability and trust in skin lesion diagnosis. *Computational and Structural Biotechnology Journal*, 28:71–79, 2025. doi: 10.1016/j.csbj.2025.02.013.

Jiao Qu, Yang Liu, Mengmei Zhang, Huan Xie, Juan Yang, Mengyuan Xu, Ling Yang, Wenlei Qian, Yan Li, Jing Jiang, Lin Zhou, Jiaxin Zeng, Danyang Tu, Jiajin Zhang, Jiachen Sun, Juan Huang, Jing Jing, Hubing Shi, Shi Gu, and Su Lui. Multiparameter concept-based interpretable model for early breast cancer diagnosis and structured reporting: a multi-center, multi-reader, radiologist-in-the-loop study. *BMC Medicine*, 24:345, 2026. doi: 10.1186/s12916-026-04889-7.

Yoshihide Sawada and Keigo Nakamura. Concept bottleneck model with additional unsupervised concepts, 2022. URL https://arxiv.org/abs/2202.01459.

Sungbin Shin, Yohan Jo, Sungsoo Ahn, and Namhoon Lee. A closer look at the intervention procedure of concept bottleneck models, 2023. URL https://arxiv.org/abs/2302.14260.

Congzheng Song and Vitaly Shmatikov. Overlearning reveals sensitive attributes. In *International Conference on Learning Representations*, 2020.

Ao Sun, Yuanyuan Yuan, Pingchuan Ma, and Shuai Wang. Eliminating information leakage in hard concept bottleneck models with supervised, hierarchical concept learning, 2024. URL https://arxiv.org/abs/2402.05945.

Moritz Vandenhirtz, Sonia Laguna, Ričards Marcinkevičs, and Julia E. Vogt. Stochastic concept bottleneck models. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

Hongmei Wang, Junlin Hou, and Hao Chen. Concept complement bottleneck model for interpretable medical image diagnosis, 2024. URL https://arxiv.org/abs/2410.15446.

Xuan Wang, Zhongling Xu, Gopi Kannedhara, Joakim Nguyen, Jian Yu, Jinrui Fang, Abdurrahmaan Baghdadi, Tianlong Chen, Awais Naeem, Chandra Krishnan, Edward Castillo, Andrew H. Song, Ankita Shukla, Ying Ding, Nicholas Konz, and Hairong Wang. ConceptM^3oE: Concept-guided multimodal mixture of experts for interpretable computational pathology, 2026.

Yongqin Xian, Christoph H. Lampert, Bernt Schiele, and Zeynep Akata. Zero-shot learning – a comprehensive evaluation of the good, the bad and the ugly, 2020. URL https://arxiv.org/abs/1707.00600.

Mateo Espinosa Zarlenga. In defense of information leakage in concept-based models, 2026. URL https://arxiv.org/abs/2606.10669.

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, 2022. URL https://arxiv.org/abs/2209.09056.

Mateo Espinosa Zarlenga, Katherine M. Collins, Krishnamurthy Dvijotham, Adrian Weller, Zohreh Shams, and Mateja Jamnik. Learning to receive help: Intervention-aware concept embedding models, 2024. URL https://arxiv.org/abs/2309.16928.

A CBM TRAINING REGIMES

Following Koh et al. (2020), we define the two additional techniques alongside joint training for training a given training concept bottleneck model (g_θ, s, f_ϕ). $\mathcal{L}_c$ sums a binary cross-entropy over the k concepts and $\mathcal{L}_y$ is the task cross entropy.

Independent. The encoder and task head are trained separately, with the task head fit to the ground-truth concepts:

$$\theta^* = \arg\min_\theta \sum_n \mathcal{L}_c\big(g_\theta(x^{(n)}), c^{(n)}\big), \quad \phi^* = \arg\min_\phi \sum_n \mathcal{L}_y\big(f_\phi(c^{(n)}), y^{(n)}\big). \tag{7}$$

At test-time the task head receives the predicted concepts $g_{\theta^*}(x)$. Independently trained CBMs where the task head receives binarised concept probabilities at test-time are refered to as Hard CBMs

Sequential. The encoder is trained as above, however the task head now receives the frozen encoder's concept predictions:

$$\theta^* = \arg\min_\theta \sum_n \mathcal{L}_c\big(g_\theta(x^{(n)}), c^{(n)}\big), \quad \phi^* = \arg\min_\phi \sum_n \mathcal{L}_y\big(f_\phi(g^*_\theta(x^{(n)})), y^{(n)}\big). \tag{8}$$

B APPLYING GC TO OTHER CONCEPT-BASED MODELS

In this section we provide details on how to apply our GC framework to several prominent concept-based models. In each case, we identify the bottleneck at which the GRL is inserted and any additional interactions with architecture-specific components.

B.1 INTERVENTION-AWARE CEMS (INTCEM)

Zarlenga et al. (2024) introduce an intervention policy network ψ that is trained jointly alongside the encoder and the task head to predicts which concepts a user should correct. The concept bottleneck remains unchanged in this architecture and so the GRL is inserted exactly as should be in GC-CEM, after the bottleneck and before the task head. The policy network remains unaffected, it continues to receives its own training signal with the GC mechanism now allowing for more faithful concepts to optimise the intervention policy on. We note that both GC training and the intervention aware policy directly improve intervenability and the combination is a natural direction, though we do not evaluate it here.

B.2 VARIATIONAL CEMS (V-CEM)

Variational CEMs (De Santis et al., 2026) map concept predictions $\hat{c}$ to concept embeddings through an approximate posterior $q(\hat{c} \mid x, c)$ which are in turn used for label prediction. V-CEMs are trained to minimise the negative ELBO which can be decomposed into a task loss L_t and a concept loss L_c. A third loss term L_p is introduced as a prior matching (KL) term.

To allow for GC training, the GRL is again placed after the concept embeddings and before the task head to cancel the task gradient at the concept encoder. The resulting concept encoder update is:

$$\nabla_\theta \mathcal{L}_{\text{GC-V-CEM}} = \lambda_c \nabla_\theta \mathcal{L}_c + \lambda_p \nabla_\theta \mathcal{L}_p. \tag{9}$$

The GRL mechanism preserves the prior-matching regulariser since this acts on the encoder directly rather than through the task head. Since the prior-matching weight λ_p interpolates V-CEM between CEM-like ($\lambda_p = 0$) and CBM-like ($\lambda_p \to \infty$) behaviour, GC removes task-driven leakage across this entire family.

B.3 HIERARCHICAL CEMS (HICEM)

HiCEM (Hill et al., 2026) extend the CEM paradigm by representing each top-level concept through positive and negative representations that additionally encode information about its sub-concepts, through sub-concept modules. The task head however still receives a standard CEM bottleneck of concatenated mixed concept embeddings. HiCEM training can be modified to be GC training by placing a GRL mechanism at between the concept bottleneck and the task head, the hierarchical structure remains unaffected since it is contained with the concept encoder g_θ.

B.4 CONCEPT BOTTLENECK MODELS WITH ADDITIONAL UNSUPERVISED CONCEPTS (CBM-AUC)

CBM-AUC (Mahinpei et al., 2021; Havasi et al., 2022; Sawada & Nakamura, 2022) augments the concept bottleneck with an unsupervised side channel to recover task accuracy under incomplete concept sets. CBM-AUCs learn unsupervised concepts through self-explaining networks and the task loss propagates through both supervised and unsupervised channels. Given the task gradient reaches supervised concepts, the leakage mechanism of Section 3.1. GC can be applied directly to the **supervised concept block** to allow for faithful and intervenable concept representations. The side channel remains unaffected learning purely from residual task information. The GC mechanism therefore strengthens this model class by allowing for low leakage concept representations whilst maintaining the accuracy benefit of the unsupervised side-channel.

B.5 STOCHASTIC CONCEPT BOTTLENECK MODELS (SCBMS)

Vandenhirtz et al. (2024) model concept logits with a multivariate Gaussian distribution, enabling interventions on one concept to update predictions of correlated concepts. Their jointly trained formulation passes sampled binary concepts to the task head and uses a straight-through estimator to propagate task gradients to the concept predictor. The objective combines concept negative log-likelihood, task loss weighted by λ_1, and precision-matrix regularisation weighted by λ_2.

GC can be applied after the sampled concept bottleneck and before the task head. Both branches must reuse the same samples and task-head computation to ensure exact cancellation of the realised task-gradient contributions. At full cancellation ($\lambda = 1$), the task loss no longer updates the concept predictor's mean and covariance parameters, which instead learn through concept likelihood and the original precision regularisation. The task head continues to train concurrently on the sampled concepts. This preserves the model's capacity to learn concept dependencies and propagate interventions, while removing task-driven pressure on the concept distribution.

C CONCEPT INCOMPLETENESS AND ITS LINK TO MI MAXIMISATION

In this section we provide a theoretical link between concept incompleteness, a condition that is known to increase leakage, and the gradient driven mechanism we outline that occurs during training.

Setup. Let $C_k = (c_1, \ldots, c_k)$ denote the set of k annotated concepts, and let $\hat{C}_k = g_\theta(x)$ denote the bottleneck. C_k is *complete* when it accounts for all task-relevant information in the input, and *incomplete* otherwise.

We make two assumptions. First, as in Section 3.1, the task head converges to the optimal predictor $f^\star_\phi(\hat{c}) = p(. \mid \hat{C}_k = \hat{c})$, under which the task loss attains the conditional entropy $\mathbb{E}[\mathcal{L}_y] = H(Y \mid \hat{C}_k) = H(Y) - I(\hat{C}_k; Y)$ and the encoder gradient therefore contains the mutual information maximisation term $-\nabla_\theta I(\hat{C}_k; Y)$ of equation 2. Second, the annotated concepts are recoverable from the input, $I(C_k; Y \mid X) = 0$, which holds whenever C_k is a deterministic function of X — as it is by construction in TabularToy.

Decomposing the task term. Applying the chain rule to $I(\hat{C}_k, C_k;\, Y)$ in both orders gives

$$I(\hat{C}_k, C_k;\, Y) \;=\; I(C_k; Y) + I(\hat{C}_k; Y \mid C_k) \;=\; I(\hat{C}_k; Y) + I(C_k; Y \mid \hat{C}_k). \tag{10}$$

When concept supervision is satisfied, C_k carries no additional information about Y beyond what $\hat{C}_k$ already provides and so $I(C_k; Y \mid \hat{C}_k) = 0$. Rearranging the right-hand equality therefore gives:

$$I(\hat{C}_k; Y) \;=\; \underbrace{I(C_k; Y)}_{} \;+\; \underbrace{I(\hat{C}_k; Y \mid C_k)}_{=:\, S(\theta)}. \tag{11}$$

The first term is fixed by the data distribution and the choice of annotated concept set; it does not depend on θ. We refer to the second term $S(\theta)$ as the *task-driven leakage signal*: it is the task information carried by the bottleneck that the ground-truth annotated concepts do not account for.

The task gradient acts only on S. Differentiating equation 11 with respect to the encoder parameters and using $\nabla_\theta I(C_k; Y) = 0$,

$$\nabla_\theta I(\hat{C}_k; Y) \;=\; \nabla_\theta S(\theta). \tag{12}$$

The mutual information maximisation term in equation 2 therefore has no concept-mediated component: whatever increase in $I(\hat{C}_k; Y)$ it produces is an increase in S. We emphasise that equation 12 characterises where this particular gradient acts, and does not assert that S is the only source of leakage in a trained model.

Incompleteness bounds S. Define the incompleteness of the annotated concept set as

$$\Delta(C_k) \;=\; I(X; Y \mid C_k) \;=\; I(X; Y) - I(C_k; Y), \tag{13}$$

the second equality following from the recoverability assumption. We note that $\hat{C}_k = g_\theta(X)$ is a deterministic function of the input X, and due to our recoverability assumption, C_k is also a deterministic function of X. Conditioning on X therefore gives $\hat{C}_k$ is independent of Y:

$$P(\hat{C}_k \mid X, C_k, Y) = P(\hat{C}_k \mid C_k, X) = P(\hat{C}_k \mid X).$$

We attain the right-hand equality since conditioning on (X, C_k) is the same as conditioning on X alone as C_k adds no new information once X is known. We therefore have the following conditional independence:

$$Y \perp \hat{C}_k \mid (X, C_k).$$

Given this relationship the data processing inequality can be applied conditionally on C_k to give:

$$S(\theta) = I(\hat{C}_k; Y \mid C_k) \;\le\; I(X; Y \mid C_k) = \Delta(C_k) \qquad \text{for every } \theta, \tag{14}$$

with equality when the encoder is task-optimal. Moreover Δ is monotone in annotation: writing $C_k^+ = C_k \cup \{c_{k+1}\}$ for the set augmented by an additional annotated concept, the chain rule gives $I(C_k^+; Y) = I(C_k; Y) + I(c_{k+1}; Y \mid C_k) \geq I(C_k; Y)$, and subtracting from the fixed quantity $I(X; Y)$ yields $\Delta(C_k^+) \leq \Delta(C_k)$, with equality precisely when c_{k+1} carries no task information beyond C_k. Annotating further concepts therefore weakly decreases Δ, which vanishes when C_k is complete.

Consequence. The maximum task information at the bottleneck is capped at $I(X; Y)$ regardless of annotation. A less complete annotated set decreases the concept-mediated component $I(C_k; Y)$ whilst leaving that cap unchanged, and by equation 14 admits a correspondingly larger task-driven leakage signal. Δ is an upper bound on $S(\theta)$, and incompleteness raises that bound. Under gradient cancellation $\nabla_\theta S = 0$ for every Δ (Proposition 1), so GC training is unaffected by this mechanism at any level of incompleteness.

Remark 1. *S captures leakage attributable to the task gradient at the encoder. Concept representations may also carry excess information through channels that receive no task gradient — most notably the soft concept probabilities passed to the task head, whose calibration encodes distributional structure beyond the binary concept label. The analysis above should not be read as predicting zero measured leakage when $\Delta = 0$.*

D Adversarial Training Formulation

In this section we provide a more complete formulation of the adversarial training setup, how it reduces leakage and why it motivates gradient cancellation. We begin by formulating a value function where we introduce an adversarial task head h_ψ alongside the main task head f_ϕ and concept encoder g_θ:

$$\begin{aligned} V(g_\theta, f_\phi, h_\psi) = \mathbb{E}_{(x,y)\sim p_{\text{data}}} \left[-\sum_j y_j \log f_\phi(g_\theta(x))_j + \lambda \sum_j y_j \log h_\psi(g_\theta(x))_j \right] \\ + \lambda_c \mathbb{E}_{(x,c)\sim p_{\text{data}}} \left[-\sum_k c_k \log g_\theta(x)_k \right]. \end{aligned} \tag{15}$$

We can decompose this into two separate optimisation problems. In the inner optimisation, the task head f_ϕ and adversary h_ψ are trained to convergence given the current concept encoder g_θ:

$$f_\phi^* = \arg\min_\phi \mathbb{E}_{(x,y)} \left[-\sum_j y_j \log f_\phi(g_\theta(x))_j \right], \quad h_\psi^* = \arg\min_\psi \mathbb{E}_{(x,y)} \left[-\sum_j y_j \log h_\psi(g_\theta(x))_j \right]. \tag{16}$$

Since they have the same optimisation problem, they converge to the same conditional distribution $f_{\phi^*}(\cdot \mid \hat{c}) = h_\psi^*(\cdot \mid \hat{c}) = p(\cdot \mid \hat{c})$.

In the outer optimisation, the encoder updated given f_ϕ^* and h_ψ^*:

$$g_\theta^* = \arg\min_\theta V(g_\theta, f_\phi^*, h_\psi^*) \tag{17}$$

During the outer concept encoder update, f_ϕ^* provides a gradient update that minimises cross entropy with respect to y whilst h_ψ^* contributes a gradient update that maximises the same cross entropy. Under the assumption that $f_\phi^* = h_\psi^*$, these two gradients are equal and opposite resulting in cancellation or removal of any task related gradient update. The encoder parameters θ are therefore updated only by concept supervision when $\lambda = 1$:

$$\nabla_\theta V(g_\theta, f_{\phi^*}, h_{\psi^*}) = \underbrace{\nabla_\theta \mathcal{L}_y\big(f_{\phi^*}(g_\theta(x)), y\big)}_{\text{task head}} - \underbrace{\lambda\, \nabla_\theta \mathcal{L}_y\big(h_{\psi^*}(g_\theta(x)), y\big)}_{\text{adversary}} + \lambda_c \nabla_\theta \mathcal{L}_c\big(s(g_\theta(x)), c\big)$$
$$= \lambda_c \nabla_\theta \mathcal{L}_c\big(s(g_\theta(x)), c\big) \quad \text{when } \lambda = 1. \tag{18}$$

In practice, this formulation requires h_ψ to track f_ϕ throughout training and any update step where cancellation doesn't occur may result in leakage or destruction of learned concept representations. We observe this instability empirically through oscillating losses and drops in concept accuracy when the adversary lags behind the task head. Moreover, we can attempt to encourage convergence of f_ϕ and h_ψ at each update step through more frequent update steps or higher learning rates for both task heads. However, these approaches don't guarantee exact cancellation. This motivates a cleaner setup where we explicitly enforce $f_\phi = h_\psi$ rather than relying on convergence, which we achieve via a GRL mechanism.

E Gradient Cancellation Reduces Leakage

We show that under the GC framework with $\lambda = 1$, the encoder is trained on concept supervision alone and therefore produces concept representations with reduced leakage. We treat CBMs and CEMs in turn. Throughout, g_θ denotes the concept encoder, f_ϕ the task head, and we use the GRL formulation in which the adversarial branch is the task head f_ϕ itself with its gradient reversed at the bottleneck. In each case we first give the general argument — that any task-driven gradient on the encoder introduces leakage irrespective of task head convergence — and then a sharper characterisation under the additional assumption of task head convergence.

E.1 CBM Case

General argument. Under vanilla joint training, the encoder gradient combines a task-driven term and a concept-supervision term:

$$\nabla_\theta \mathcal{L}_{\text{joint}} = \underbrace{\nabla_\theta \mathcal{L}_y(f_\phi(g_\theta(x)), y)}_{\text{task-driven}} + \lambda_c \nabla_\theta \mathcal{L}_c(g_\theta(x), c). \tag{19}$$

The task-driven term updates the encoder to make the concept representation $\hat{C}$ more predictive of the task label, exploiting any feature of the input — including spurious features not captured by the concept ontology. This is the general source of leakage, and it acts at every training step irrespective of the task head's state. The encoder is not trained on concept supervision alone, so its representations need not align with the concept labels.

Characterisation under task head convergence. We can characterise the task-driven term more precisely under the additional assumption that the task head converges to the optimal predictor $f_\phi^* = p(y \mid \hat{c})$. In this case the task loss attains the conditional entropy $\mathbb{E}[\mathcal{L}_y] = H(Y \mid \hat{C})$, and using the identity $H(Y \mid \hat{C}) = H(Y) - I(\hat{C}; Y)$ with $\nabla_\theta H(Y) = 0$, the task-driven term becomes an explicit mutual information maximisation:

$$\nabla_\theta \mathcal{L}_{\text{joint}} = -\nabla_\theta I(\hat{C}; Y) + \lambda_c \nabla_\theta \mathcal{L}_c(g_\theta(x), c). \tag{20}$$

The encoder maximises $I(\hat{C}; Y)$ by encoding spurious task-correlated features. Mutual information maximisation is therefore a special case of the general task-driven leakage mechanism, holding when the task head has converged.

Gradient cancellation. Under the GC framework, the encoder receives two task gradients through the shared task head — one standard and one reversed by the GRL — which combine as $\nabla_\theta \mathcal{L}_y - \lambda \nabla_\theta \mathcal{L}_y$. At $\lambda = 1$, the task-driven term is cancelled exactly at the encoder at every training step, *independent of task head convergence*:

$$\nabla_\theta \mathcal{L}_{\text{GC}} = \lambda_c \nabla_\theta \mathcal{L}_c(g_\theta(x), c). \tag{21}$$

The encoder is now trained on concept supervision alone and, at the stationary point, recovers the true concept posterior:

$$g_{\theta^*}(x) = \arg\min_g \mathbb{E}_{(x,c)} \left[\mathcal{L}_c(g(x), c)\right] = p(c \mid x). \tag{22}$$

Leakage is reduced since the encoder learns concepts faithfully from concept supervision alone, with no task-driven pressure to encode features outside the concept ontology.

E.2 CEM CASE

For CEMs the bottleneck is the concatenated mixed embedding $\hat{c} = [\hat{c}_1, \ldots, \hat{c}_i]$, where each concept embedding is a linear combination of positive and negative prototypes weighted by the predicted probability:

$$\hat{c}_i = \hat{p}_i \rho_i^+ + (1 - \hat{p}_i)\rho_i^-, \qquad \hat{p}_i = s_i\left([\rho_i^+, \rho_i^-]\right). \tag{23}$$

The encoder output is the tuple $(\hat{p}, \hat{c})$ where $\hat{p}$ are the concept probabilities used for supervision and $\hat{c}$ the embeddings passed to the task head. Concept supervision acts on $\hat{p}$.

General argument. As in the CBM case, vanilla joint training applies a task-driven gradient to the encoder through the embedding $\hat{c}$. This term shapes both the predicted probabilities $\hat{p}$ and the concept prototypes $\rho_i^{\pm}$ to increase task predictiveness, irrespective of task head convergence. We show that GC removes the task-driven gradient at both levels.

Characterisation under task head convergence. When the task head converges to $f_\phi^* = p(y \mid \hat{c})$, the effective encoder objective takes the same form as the CBM case:

$$V(g_\theta, f_\phi^*) = V_g = (1 - \lambda)[H(Y) - I(\hat{C}; Y)] + \lambda_c \mathcal{L}_c(\hat{p}, c), \tag{24}$$

allowing us to characterise the task-driven term as mutual information maximisation, as before. We use this form to derive the optimal probabilities and embeddings.

Probability level. Taking the gradient with respect to $\hat{p}_i$ and applying the chain rule through $\hat{c}_i$:

$$\nabla_{\hat{p}_i} V_g = (\lambda - 1)(\rho_i^+ - \rho_i^-) \cdot \nabla_{\hat{c}_i} I(\hat{C}; Y) + \lambda_c \nabla_{\hat{p}_i} \mathcal{L}_c(\hat{p}_i, c_i). \tag{25}$$

At $\lambda = 1$ the mutual information term vanishes:

$$\nabla_{\hat{p}_i} V_g = \lambda_c \nabla_{\hat{p}_i} \mathcal{L}_c(\hat{p}_i, c_i). \tag{26}$$

Setting this to zero with $\mathcal{L}_c$ the binary cross-entropy yields the equilibrium condition:

$$\frac{\hat{p}_i^* - c_i}{\hat{p}_i^*(1 - \hat{p}_i^*)} = 0 \implies \hat{p}_i^* = p(c_i \mid x). \tag{27}$$

The predicted probabilities converge to the true concept posteriors,

Vector level. We analyse the positive and negative prototypes separately. The embedding $\hat{c}_i$ depends on ρ_i^+ both directly through the mixture and indirectly through the predicted probability $\hat{p}_i = s_i([\rho_i^+, \rho_i^-])$. Applying the chain rule with both paths:

$$\nabla_{\rho_i^+} V_g = \frac{\partial V_g}{\partial \hat{c}_i} \frac{\partial \hat{c}_i}{\partial \rho_i^+} + \frac{\partial V_g}{\partial \hat{p}_i} \frac{\partial \hat{p}_i}{\partial \rho_i^+}. \tag{28}$$

The first path captures the direct dependence of the mixture on ρ_i^+, giving $\partial \hat{c}_i / \partial \rho_i^+ = \hat{p}_i$. The second path captures the dependence through the scoring function, where the concept supervision contributes $\partial V_g / \partial \hat{p}_i = \lambda_c \nabla_{\hat{p}_i} \mathcal{L}_c(\hat{p}_i, c_i)$. Substituting both:

$$\nabla_{\rho_i^+} V_g = \hat{p}_i \frac{\partial V_g}{\partial \hat{c}_i} + \lambda_c \nabla_{\hat{p}_i} \mathcal{L}_c(\hat{p}_i, c_i)\, \nabla_{\rho_i^+} s(\rho_i^+). \tag{29}$$

The scoring function applies a sigmoid to a linear map of the concatenated prototypes. Splitting the scoring weights as $W_s = [w_{s+}, w_{s-}]$, the pre-activation is $W_s[\rho_i^+, \rho_i^-]^\top = w_{s+}\rho_i^+ + w_{s-}\rho_i^-$, so the predicted probability is:

$$\hat{p}_i = \sigma\left(w_{s+}\rho_i^+ + w_{s-}\rho_i^-\right). \tag{30}$$

Using the sigmoid derivative $\sigma'(z) = \sigma(z)(1-\sigma(z)) = \hat{p}_i(1-\hat{p}_i)$ and applying the chain rule with respect to each prototype:

$$\nabla_{\rho_i^+} s = \hat{p}_i(1-\hat{p}_i)\, w_{s+}, \qquad \nabla_{\rho_i^-} s = \hat{p}_i(1-\hat{p}_i)\, w_{s-}. \tag{31}$$

The cross-entropy gradient takes the prediction-error form $\nabla_{\hat{p}_i} \mathcal{L}_c = (\hat{p}_i - c_i)/[\hat{p}_i(1-\hat{p}_i)]$. Substituting these alongside the mutual information term for $\partial V_g / \partial \hat{c}_i$, the $\hat{p}_i(1-\hat{p}_i)$ factors cancel and we obtain:

$$\nabla_{\rho_i^+} V_g = \hat{p}_i(\lambda - 1)\nabla_{\hat{c}_i} I(\hat{C}; Y) + \lambda_c(\hat{p}_i - c_i) w_{s+} \tag{32}$$

$$\nabla_{\rho_i^-} V_g = (1-\hat{p}_i)(\lambda - 1)\nabla_{\hat{c}_i} I(\hat{C}; Y) + \lambda_c(\hat{p}_i - c_i) w_{s-} \tag{33}$$

At $\lambda = 1$ the mutual information term vanishes from both prototype gradients:

$$\nabla_{\rho_i^+} V_g = \lambda_c(\hat{p}_i - c_i) w_{s+} = \alpha w_{s+}, \qquad \nabla_{\rho_i^-} V_g = \lambda_c(\hat{p}_i - c_i) w_{s-} = \alpha w_{s-}. \tag{34}$$

Both prototypes are updated solely through concept supervision — their update directions are parallel to the scoring weights w_{s+} and w_{s-} respectively, scaled by the concept prediction error. Given that both the concept prototypes and probabilities are updated on concept supervision alone and the mixed embedding is a linear combination of them, the mixed concept vector is also updated through concept supervision alone. We therefore remove this task-based mutual information term that results in leakage.

Contrast with vanilla CEM. For completeness, at $\lambda = 0$ the prototype gradients retain the mutual information term:

$$\nabla_{\rho_i^+} V_g = -\hat{p}_i \nabla_{\hat{c}_i} I(\hat{C}; Y) + \lambda_c(\hat{p}_i - c_i) w_{s+} \tag{35}$$

$$\nabla_{\rho_i^-} V_g = (\hat{p}_i - 1)\nabla_{\hat{c}_i} I(\hat{C}; Y) + \lambda_c(\hat{p}_i - c_i) w_{s-} \tag{36}$$

The prototypes are updated by two competing forces — concept supervision and task-driven mutual information maximisation — and therefore do not converge to concept-pure prototypes.

We note that the mutual information form of the task-driven term holds under the assumption of task head convergence ($f_\phi^* = p(y \mid \hat{c})$), which we adopt throughout this derivation to make the connection between the task gradient and leakage explicit. The argument does not depend on this assumption: without convergence the mutual information term is simply replaced by the general task-driven gradient $\hat{p}_i\, \partial \mathcal{L}_y / \partial \hat{c}_i$, which is likewise cancelled at $\lambda = 1$ and prevents concept-pure prototypes when present. Task head convergence is not required for the leakage mechanism or its elimination.

F Toy experiment for leakage control

In this section we define a controlled gaussian dataset and provide a toy experiment demonstrating how varying λ affects leakage,

F.1 Controlled Gaussian Dataset

We construct a synthetic dataset with $k = 6$ binary concepts drawn i.i.d:

$$c_i \sim \text{Bernoulli}(0.5), \qquad i = 1, ..., k$$

The task label is a deterministic function of a weighted concept score

$$s = \sum_{i=1}^{k} w_i c_i, \qquad w_i \sim \mathcal{N}(0,1) \text{ i.i.d.}$$

discretised into L equal-frequency quantile bins to obtain $y \in \{0, \ldots, L-1\}$. For the binary case ($L = 2$) we set $w_i = 1$ for simplicity, recovering the majority-vote label $y = \mathbb{1}\{s \geq K/2\}$; for $L > 2$ we draw w_i randomly to ensure s takes continuous, non-repeating values across the 2^k concept patterns, avoiding bin collisions at large k.

By construction y is a function of the concepts alone and provides no information beyond what the concepts already contain — there is no residual task information unless we explicitly inject it.

Each concept c_i is assigned a concept embedding $c_i \in \mathbb{R}^d$ via:

$$c_i = c_i \cdot \mu_i^+ + (1 - c_i) \cdot \mu_i^- + \eta + \epsilon,$$

where $\mu_i^+, \mu_i^- \in \mathbb{R}^d$ are fixed prototype vectors drawn i.i.d from $N(0, \mathbf{I}_d)$ using a fixed random seed. $\epsilon \sim N(0, \mathbf{I}_d)$ is the independent noise and η is the leakage term that introduces both concept-task leakage $\alpha\phi_y$ and interconcept leakage $\beta\psi(c_o)$. The clean embedding is therefore a gaussian mixture that is linearly separable in c_i but contains no additional information about y or additional interconcept information. We use $N = 3,000$ samples with a 0.66/0.33 train test split and average all configurations over 20 folds to report mean and standard deviation.

Concept-Task Leakage. To simulate concept-task leakage, we add a class-conditional offset to each concept embedding:

$$c_i \leftarrow c_i + \alpha\phi_y,$$

where $\phi_y \sim N(0, \mathbf{I}_d)$ is a fixed vector drawn per class (i.e in binary case: ϕ_0 and ϕ_1) and is shared across all concepts. $\alpha \in [0, 3]$ is the injected leakage strength. As α increases the task label y is increasingly encoded into the residual structure of concept embeddings, we would therefore expect RTL to rise monotonically. At $\alpha = 0$ concept embeddings are a clean Gaussian mixture and so residuals after projecting out c_i should be pure noise, the RTL in this case should be zero.

Interconcept Leakage. To simulate interconcept leakage, more specifically additional interconcept information regarding concept c_j in c_i we inject concept c_j's label into c_i's embedding:

$$c_i \leftarrow c_i + \beta\psi(c_j),$$

$\psi \in \mathbb{R}^d$ is a fixed vector and $\beta \in [0, 3]$ is the leakage strength. All other concept embeddings remain unaffected by this injection. This is a directed injection affecting only the interconcept relationship between concept i and j. We expect RCL to monotonically increase as β increases.

F.2 λ AS A CONTROL FOR LEAKAGE

We note that our adversarial implementation of $\lambda = 1$ produces identical gradients to a stop-gradient mechanism at the concept bottleneck. However our adversarial setup allows for a explicit and continuously tunable parameter with a clear interpretation at each end. At $\lambda = 0$ we have a standard jointly trained concept bottleneck model and at $\lambda = 1$ we have a task head trained detached from the concept encoder. Although, increasing λ_c is also known to decrease leakage, there is no clear interpretation of an arbitrarily large value which may also be dataset dependent.

We validate the claim that our λ acts as a tunable control on how much task-driven leakage the concept encoder admits by sweeping λ from 0 to 1 in increments of 0.1 and tracking RTL and RCL. We use our toy gaussian dataset, train toy GC-CEM models with layer sizes $\{96, 32, 192\}$ and a linear task head. We set $\lambda_c = 0.1$ and $\alpha = \beta = 1.5$:

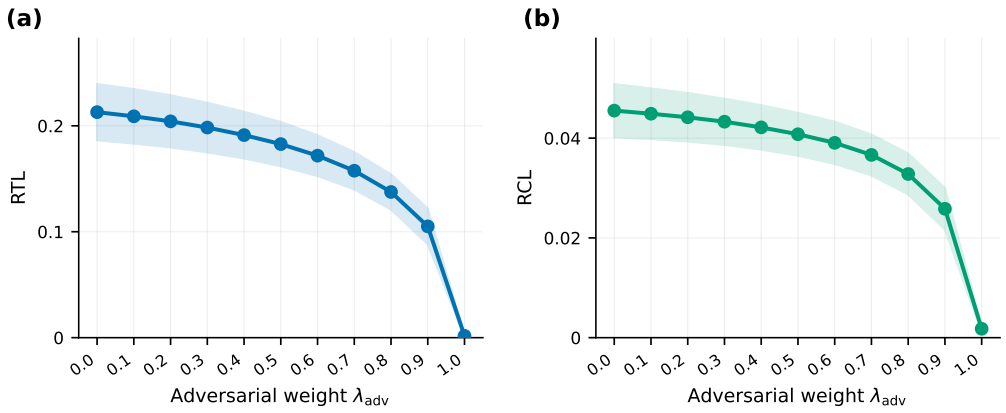


Figure 8: Validation of λ as a control for task-driven leakage.

From Figure 8, we observe both metrics decrease monotonically in λ and hit near zero at $\lambda = 1$. We note the curve is not linear and there is a sharp drop in leakage from $\lambda = 0.9$ to $\lambda = 1$, which points towards partial cancellation providing a weaker leakage reduction when compared to full cancellation.

G Proofs of Theoretical Results

G.1 Proof of Proposition 1

Proof. In a Sequential CBM, the encoder parameters θ are updated solely by the concept supervision:

$$\nabla_\theta V_{seq} = \nabla_\theta \mathcal{L}_c(g_\theta(x), c). \tag{37}$$

In the GC-CBM, the total objective V is optimized jointly. However, the Gradient Reversal Layer (GRL) acts as a linear operator $R(\nabla\mathcal{L}) = -\lambda\nabla\mathcal{L}$ on the adversarial branch during backpropagation. The total gradient with respect to the encoder θ is:

$$\nabla_\theta V_{GC-CBM} = \lambda_c \nabla_\theta \mathcal{L}_c + \nabla_\theta \mathcal{L}_y + R(\nabla_\theta \mathcal{L}_y). \tag{38}$$

Substituting the GRL operator R and the adversarial weight $\lambda = 1$:

$$\nabla_\theta V_{GC-CBM} = \lambda_c \nabla_\theta \mathcal{L}_c + \nabla_\theta \mathcal{L}_y - \nabla_\theta \mathcal{L}_y. \tag{39}$$

Since the task head and the adversarial task head are parameterized by the same weights ϕ and receive the same input embeddings $\hat{c} = g_\theta(x)$, their loss functions and partial derivatives are identical. We therefore obtain the following:

$$\nabla_\theta V_{GC-CBM} = \lambda_c \nabla_\theta \mathcal{L}_c + \nabla_\theta \mathcal{L}_y - \nabla_\theta \mathcal{L}_y = \lambda_c \nabla_\theta \mathcal{L}_c. \tag{40}$$

As $\nabla_\theta V_{GC-CBM} \equiv \nabla_\theta V_{seq}$, the encoder in the GC-CBM converges to the same stationary point θ^* as the Sequential CBM, effectively decoupling the task supervision from the concept representation learning phase.

We now focus on the sequential CBM task head gradient, which is evaluated at the fixed optimum g_{θ^*}. We define the empirical objective $V(\phi) = \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}_y(f_\phi(g_\theta(x_i)), y_i)$, whose gradient is the average of the per-sample gradients. The empirical objective in the Sequential CBM is therefore:

$$\nabla_\phi V_{seq} = \frac{1}{N}\sum_{i=1}^{N} \nabla_\phi \mathcal{L}_y(f_\phi(g_{\theta^*}(x_i)), y_i). \tag{41}$$

In the GC-CBM, the task head gradient update is the sum of the two branch contributions, evaluated at the current encoder state θ_t. For equivalence in updates to the Sequential CBM we rescale the loss terms by a factor of $\frac{1}{2}$:

$$\nabla_\phi V_{\text{GC-CBM}} = \frac{1}{2}\Big[\nabla_\phi V_{\text{task}} + \nabla_\phi V_{\text{GRL}}\Big] \tag{42}$$

$$= \frac{1}{2N}\sum_{i=1}^{N}\Big[\nabla_\phi \mathcal{L}_y(f_\phi(g_{\theta_t}(x_i)), y_i) + \nabla_\phi \mathcal{L}_y(f_\phi(R_\lambda(g_{\theta_t}(x_i))), y_i)\Big] \tag{43}$$

$$= \frac{1}{N}\sum_{i=1}^{N}\nabla_\phi \mathcal{L}_y(f_\phi(g_{\theta_t}(x_i)), y_i). \tag{44}$$

Given we have already shown $\nabla_\theta V_{GC-CBM} = \lambda_c \nabla_\theta \mathcal{L}_c$, meaning the encoder update is independent of ϕ. Thus, $\theta_t \to \theta^*$ according to the standard convergence properties of the concept loss.

We assume the map $\hat{c} \mapsto \nabla_\phi \mathcal{L}_y(f_\phi(\hat{c}), y)$ is globally L-Lipschitz in the concept bottleneck $\hat{c}$ — a property satisfied in our setting, where f_ϕ is a linear task head and $\mathcal{L}_y$ is softmax cross-entropy with bounded logits, both of which have Lipschitz gradients on the bounded domain $[0,1]^K$. We note that for CEMs the same map is locally L-Lipschitz and not globally since the bottleneck domain is unbounded.

Applying this per sample and averaging over the training set, the error term in the gradient estimate $||\nabla_\phi V_{GC-CBM} - \nabla_\phi V_{seq}||$ is bounded by the Lipschitz constant L of the map $\hat{c} \mapsto \nabla_\phi \mathcal{L}_y(f_\phi(\hat{c}), y)$:

$$\|\nabla_\phi V_{GC-CBM} - \nabla_\phi V_{seq}\| \le \frac{L}{N}\sum_{i=1}^{N}\|g_{\theta_t}(x_i) - g_{\theta^*}(x_i)\|. \tag{45}$$

Since $\|g_{\theta_t} - g_{\theta^*}\| \to 0$ by the convergence of the encoder to stationary point θ^*, the GC-CBM task head gradient recovers the sequential gradient asymptotically. Under standard stochastic approximation assumptions (e.g. Robbins-Monro conditions on the learning rate), the task head parameters ϕ will converge to the minimum of the sequential task objective. □

G.2 Proof of Proposition 2

Proof. By Proposition 1 the encoder update direction under GC-CBM is:

$$\nabla_\theta V_{\text{GC-CBM}} = \lambda_c \nabla_\theta \mathcal{L}_c. \tag{46}$$

Assuming standard gradient descent with learning rate η, the encoder parameter update is:

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta V_{\text{GC-CBM}} \tag{47}$$

$$= \theta_t - \eta\lambda_c \nabla_\theta \mathcal{L}_c \tag{48}$$

$$= \theta_t - \tilde{\eta}\,\nabla_\theta \mathcal{L}_c, \tag{49}$$

where $\tilde{\eta} = \eta\lambda_c$ is an effective learning rate. The concept weight λ_c therefore enters the update only through the product $\eta\lambda_c$ — it rescales the step size but does not alter the update direction $\nabla_\theta \mathcal{L}_c$. Consequently, for any $\lambda_c > 0$, converges to the same stationary point θ^* defined by $\nabla_\theta \mathcal{L}_c = 0$, provided $\tilde{\eta}$ satisfies standard step-size conditions for convergence (e.g. the Robbins–Monro conditions). The task head update is solely defined by $\mathcal{L}_y$ and is independent of λ_c. □

Remark 2. *A sigmoidal CBM and vanilla CEM can approximate gradient cancellation in a limiting regime. Setting $\lambda_c \gg 1$ with learning rate $\eta = 1/\lambda_c$, the encoder update becomes*

$$\theta_{t+1} = \theta_t - \frac{1}{\lambda_c}\left[\nabla_\theta \mathcal{L}_y + \lambda_c \nabla_\theta \mathcal{L}_c\right] \tag{50}$$

$$= \theta_t - \frac{1}{\lambda_c}\nabla_\theta \mathcal{L}_y - \nabla_\theta \mathcal{L}_c, \tag{51}$$

and as $\lambda_c \to \infty$ the task term vanishes, leaving the concept-only update $\theta_{t+1} = \theta_t - \nabla_\theta \mathcal{L}_c$. However this requires an impractically large λ_c and correspondingly small learning rate, slowing convergence. Our framework achieves exact *cancellation at every step for any $\lambda_c > 0$, without this trade-off.*

H MUTUAL INFORMATION BASED LEAKAGE METRICS

We provide full mathematical descriptions of CTL and ICL in this section:

Concepts-Task Leakage. If CTL is present then the mutual information (MI) between learned concepts and task labels should be higher than the MI between the ground truth concepts and task labels.

Let $H(Z)$ denote the entropy of random variable Z and $I(Z, W)$ the MI between random variables Z and W. The CTL for concept i (CTL_i) and average CTL are defined as follows:

$$\text{CTL}_i(\hat{C}_i, C_i, Y) = \max\left(0, \frac{I(\hat{C}_i, Y) - I(C_i, Y)}{H(Y)}\right), \qquad \text{CTL}(\hat{C}, C, Y) = \frac{1}{k}\sum_{i=1}^{k} \text{CTL}_i(\hat{C}_i, C_i, Y) \tag{52}$$

Interconcept Leakage. Similarly, if ICL is present, the learned interconcept MI is higher than the ground truth interconcept MI. The accompanying measure for this type of leakage is defined as follows:

$$\text{ICL}_{ij}(\hat{C}, C) = \max\left(0, \frac{I(\hat{C}_i, \hat{C}_j)}{\sqrt{H(\hat{C}_i)H(\hat{C}_j)}} - \frac{I(C_i, C_j)}{\sqrt{H(C_i)H(C_j)}}\right) \tag{53}$$

Per-concept and average ICL are also defined below:

$$\text{ICL}_i(\hat{C}, C) = \frac{1}{k-1}\sum_{j=1}^{k} \text{ICL}_{ij}(\hat{C}, C), \qquad \text{ICL}(\hat{C}, C) = \frac{1}{k}\sum_{i=1}^{k} \text{ICL}_i(\hat{C}, C). \tag{54}$$

Both quantities measure excess information relative to a ground truth baseline and are normalised by entropy such that $0 \leq \text{ICL}, \text{CTL} \leq 1$.

I RTL AND RCL

I.1 VALIDATION OF RTL AND RCL

We perform two experiments using the dataset outlined in Appendix F.1 to validate our leakage metric.

Experiment 1: Sensitivity to leakage and robustness to dimensionality. Firstly we vary respective leakage strengths α, β and measure RTL and RCL directly on the leaky embeddings for a range of embedding dimensions $m \in \{8, 16, 32, 64, 128\}$ to ensure our metrics *detect* injected leakage - increasing monotonically with leakage strength - and that they are *dimensionality-agnostic*, exhibiting this behaviour consistently regardless of embedding size. To inject interconcept leakage at a global level and not just between a concept pair, we leak each concepts information into its neighbours i.e. c_i leaks into c_{i+1} and c_k leaks into c_1.

To be clear, for this experiment we do not train any models, we simulate leaky embeddings which could feasibly be the output from a trained CEM and apply our leakage metrics directly.

Figure 9 confirms both metrics behave as required. RTL and RCL are near zero when there is no injected leakage and rise monotonically with α and β respectively. This behaviour is observed consistently across all dimensionalities and so we conclude our metric is sensitive to leakage and dimensionality-agnostic.

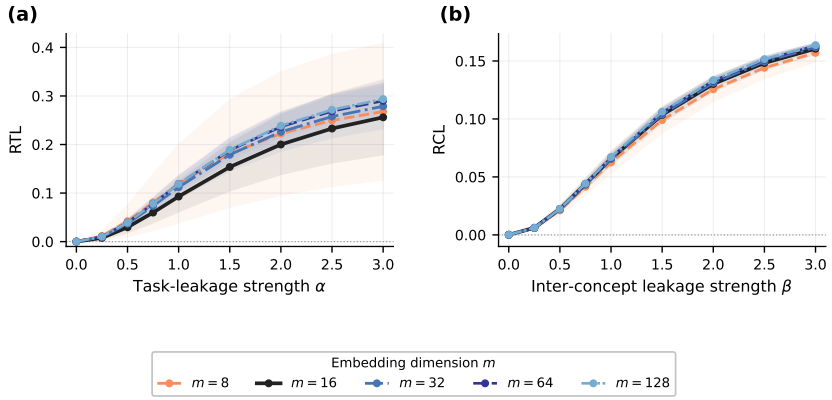


Figure 9: Validation of RTL and RCL on controlled Gaussian data. **Left:** RTL as a function of injected task-leakage strength α. **Right:** RCL as a function of injected inter-concept leakage strength β applied across neighbouring concept pairs

Experiment 2: Sensitivity to concept supervision Secondly, it is well known that decreasing concept supervision λ_c results in increased leakage (Parisini et al., 2026). We therefore expect RTL and RCL to monotonically increase as concept supervision decreases. We jointly train small toy CEM models on our controlled gaussian dataset with $\alpha = \beta = 1.5$ and vary concept supervision $\lambda_c \in \{0.05, 0.1, 0.5, 1, 5, 10\}$ to confirm that our metric is sensitive leakage as it appears in trained models.

We define our concept encoder as a two layer LeakyReLU network with layer sizes $96, 32, 192$ with a final layer sigmoid activation and a linear task head.

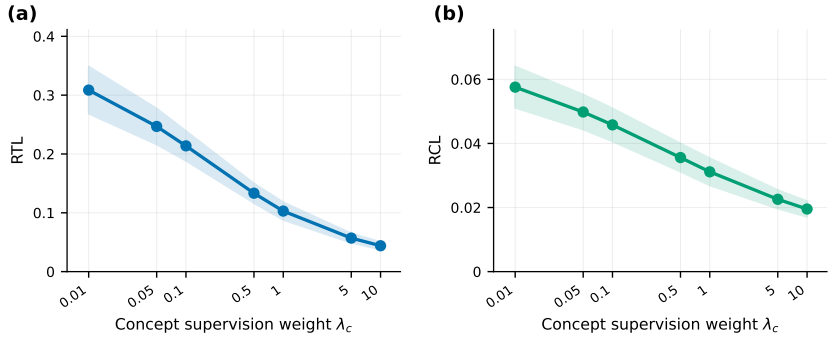


Figure 10: Validation of RTL and RCL on toy CEMs trained with increasing concept supervision weight λ_c. **Left:** RTL as a function of λ_c. **Right:** RCL as a function of λ_c

Figure 10 shows that both RTL and RCL decrease monotonically as λ_c increases. This sanity check confirms that our metrics are sensitive to leakage as it appears in trained models.

J Experimental Setup

J.1 Datasets

We evaluate all models across three datasets.

TabularToy(δ). This is a synthetic dataset introduced by Mahinpei et al. (2021) constructed through sampling a 3-dimensional latent variable $z \sim \mathcal{N}(0, \Sigma(\delta))$, with correlation matrix $\Sigma(\delta)_{ij} = \delta_{ij} + \delta(1 - \delta_{ij})$ with $i, j = 1, 2, 3$. $\delta \in (-1, 1)$ controls the correlation between components and δ_{ij} is the Kronecker symbol. The 7-dimensional input x is a trigonometric function of the latent variables and the three binary concepts are defined by the sign of the latent variables, $c_i = (z_i > 0)$. The task

label is binary and has the following functional form:

$$y_{TT} = (c_1 + c_2 + c_3 \geq 2).$$

All experiments with this dataset use 10,000 samples with a 0.7/0.2/0.1 train/validation/test split.

CUB. This dataset is made up of 11,788 images of birds which are labelled with 112 binary physical attributes (e.g. has black eyes). The dataset consists of 200 task labels each corresponding to a bird species (Wah et al., 2011). CUB is a more complex real-world dataset which also provides rich and diverse concept annotations that can almost completely determine the task label (i.e bird species). We match the format in Koh et al. (2020) and use a balanced 0.4/0.1/0.5 train/ validation / test ratio.

Animals with Attributes 2 (AwA2). This dataset consists of 37,322 animal images across 50 classes, each annotated with 85 binary attributes (Xian et al., 2020). AwA2 provides no predefined semantic concept grouping like CUB for grouped intervention procedures and so we define 17 groups for interventions (Table 1) covering appearance, behaviour and social characteristics. Interventions are performed group by group with random order.

Table 1: Semantic concept groups defined for AwA2 intervention experiments. n denotes the number of attributes per group.

Group	Name	n	Attributes
0	Colour	8	black, white, blue, brown, gray, orange, red, yellow
1	Surface covering	6	patches, spots, stripes, furry, hairless, toughskin
2	Body morphology	4	big, small, bulbous, lean
3	Anatomy	15	flippers, hands, hooves, pads, paws, longleg, longneck, tail, chewteeth, meatteeth, buckteeth, strainteeth, horns, claws, tusks
4	Locomotion	7	flys, hops, swims, tunnels, walks, bipedal, quadrapedal
5	Physical capability	6	fast, slow, strong, weak, muscle, agility
6	Activity pattern	4	active, inactive, nocturnal, hibernate
7	Diet	5	fish, meat, plankton, vegetation, insects
8	Foraging	6	forager, grazer, hunter, scavenger, skimmer, stalker
9	Biogeography	2	newworld, oldworld
10	Climate/biome	9	arctic, coastal, desert, bush, plains, forest, fields, jungle, mountains
11	Substrate	5	ocean, ground, water, tree, cave
12	Social	2	group, solitary
13	Temperament	3	fierce, timid, smart
14	Nesting	1	nestspot
15	Domestication	1	domestic
16	Other	1	smelly

J.2 Model Architectures and training

We perform a grid search for each configuration and select the learning rate that maximise validation accuracy

For TabularToy, we use a 4-layer LeakyReLU MLP with layer widths $\{7, 64, 64, 3\}$ as a concept encoder. For CUB we use a ResNet-18 pretrained on ImageNet as the concept encoder and for AwA2 we use a ResNet-34 with the same pretraining. When training embedding models we use 16-dimensional concept embeddings for TabularToy and AwA2 and 32 for CUB. We set the batchsize to 64 for all models trained on TabularToy, 256 for CUB and 512 for AwA2 to make better use of GPU memory. All results are reported as mean and standard deviation across 5 random seeds.

All models use the Adam optimiser (Kingma & Ba, 2017) and are trained on a Nvidia A100 or A40 GPU with 40GB memory.

Table 2: Table of learning rates and number of epochs across datasets and architectures for reproducibility

Model	Epochs	Learning rate
TabularToy		
Hard CBM	90	0.05
Sequential CBM	90	0.05
Joint CBM	90	0.05
GC-CBM (ours)	90	0.05
CEM	90	0.05
GC-CEM (ours)	90	0.001
AwA2		
Hard CBM	100	0.001
Sequential CBM	100	0.001
Joint CBM	120	0.001
GC-CBM (ours)	120	0.0005
CEM	90	0.001
GC-CEM (ours)	90	0.0005
CUB-200		
Hard CBM	200	0.0001
Sequential CBM	200	0.0001
Joint CBM	250	0.0001
GC-CBM (ours)	300	0.0001
CEM	300	0.001
GC-CEM (ours)	300	0.0005

We increase the number of epochs for CUB and use the best checkpoint for validated task accuracy to combat training instability. We also find GC training sometimes takes longer for convergence on CUB. Learning rates were selected from a quick pilot grid search. In certain cases GC training requires a lower learning rate for stable training. Number of epochs reported for Hard and Sequential is the number of epochs for the concept and task head **each**.

Welch's unpaired two-sided t-tests with the Holm correction (n = 5 folds per configuration) at α = 0.05 are used to check for statistical significance on all evaluation metrics.

K EXPERIMENTAL RESULTS

In this section we provide complete tables of results for all experiments:

K.1 GC-CBM RESULTS

K.1.1 TASK AND CONCEPT ACCURACY

Table 3: Task accuracy (Task) and concept accuracy (Con) for CBM variants across datasets and λ_c values. Results are mean $\pm$ std over 5 folds. **Bold** denotes the significantly higher value between GC-CBM and Joint CBM at matched λ_c (two-sided Welch t-test with Holm correction, $\alpha = 0.05$).

		TabularToy		**AwA2**		**CUB**	
Model	λ_c	Task	Con	Task	Con	Task	Con
Hard CBM	–	$99.5_{\pm 0.29}$	$99.7_{\pm 0.13}$	$83.1_{\pm 0.38}$	$96.1_{\pm 0.07}$	$61.8_{\pm 0.28}$	$94.4_{\pm 0.01}$
Sequential CBM	–	$99.6_{\pm 0.08}$	$99.5_{\pm 0.13}$	$83.1_{\pm 0.50}$	$96.1_{\pm 0.10}$	$65.9_{\pm 0.10}$	$94.7_{\pm 0.04}$
Joint CBM	0.1	$99.6_{\pm 0.18}$	$57.2_{\pm 13.91}$	$82.9_{\pm 0.61}$	$89.7_{\pm 2.87}$	$\mathbf{71.0}_{\pm 0.84}$	$71.1_{\pm 0.89}$
	0.5	$\mathbf{99.6}_{\pm 0.14}$	$94.0_{\pm 1.75}$	$83.0_{\pm 0.41}$	$95.6_{\pm 0.06}$	$\mathbf{72.4}_{\pm 1.03}$	$86.5_{\pm 0.48}$
	1	$\mathbf{99.6}_{\pm 0.13}$	$96.5_{\pm 0.64}$	$83.0_{\pm 0.62}$	$95.8_{\pm 0.17}$	$\mathbf{68.0}_{\pm 0.58}$	$91.2_{\pm 0.04}$
GC-CBM	0.1	$99.3_{\pm 0.38}$	$\mathbf{99.5}_{\pm 0.34}$	$\mathbf{85.2}_{\pm 0.74}$	$\mathbf{96.5}_{\pm 0.17}$	$65.9_{\pm 0.16}$	$\mathbf{94.7}_{\pm 0.02}$
	0.5	$99.3_{\pm 0.08}$	$\mathbf{99.5}_{\pm 0.13}$	$\mathbf{85.0}_{\pm 0.65}$	$\mathbf{96.5}_{\pm 0.16}$	$66.1_{\pm 0.76}$	$\mathbf{94.6}_{\pm 0.10}$
	1	$99.3_{\pm 0.19}$	$\mathbf{99.5}_{\pm 0.28}$	$\mathbf{84.9}_{\pm 0.87}$	$\mathbf{96.5}_{\pm 0.22}$	$66.5_{\pm 0.24}$	$\mathbf{94.8}_{\pm 0.06}$

K.1.2 CONCEPT-TASK LEAKAGE AND INTERCONCEPT LEAKAGE

Table 4: **Concepts–Task Leakage (CTL) and Interconcept Leakage (ICL).** Mean $\pm$ std over 5 folds. **Bold** denotes the significantly lower value between Joint CBM and GC-CBM at matched λ_c (two-sided Welch t-test with Holm correction $\alpha = 0.05$). Hard and Sequential CBM are architectural reference points, not λ_c-matched. CUB standard deviation values are given to 4 d.p.

		TabularToy		**AwA2**		**CUB**	
Model	λ_c	CTL	ICL	CTL	ICL	CTL	ICL
Hard CBM	–	$0.065_{\pm .041}$	$0.055_{\pm .051}$	$0.026_{\pm .004}$	$0.027_{\pm .007}$	$0.034_{\pm .0001}$	$0.005_{\pm .0001}$
Sequential CBM	–	$0.058_{\pm .036}$	$0.046_{\pm .040}$	$0.029_{\pm .011}$	$0.026_{\pm .008}$	$0.030_{\pm .0009}$	$0.008_{\pm .0004}$
Joint CBM	0.1	$0.729_{\pm .004}$	$0.337_{\pm .054}$	$0.113_{\pm .022}$	$0.021_{\pm .005}$	$0.043_{\pm .0009}$	$0.009_{\pm .0004}$
	0.5	$0.658_{\pm .043}$	$0.268_{\pm .036}$	$0.050_{\pm .020}$	$0.023_{\pm .004}$	$0.044_{\pm .0037}$	$0.012_{\pm .0014}$
	1	$0.581_{\pm .068}$	$0.213_{\pm .027}$	$0.031_{\pm .012}$	$0.025_{\pm .009}$	$0.032_{\pm .0024}$	$0.013_{\pm .0005}$
GC-CBM	0.1	$\mathbf{0.089}_{\pm .045}$	$\mathbf{0.053}_{\pm .031}$	$0.094_{\pm .014}$	$\mathbf{0.010}_{\pm .001}$	$\mathbf{0.024}_{\pm .0006}$	$0.010_{\pm .0001}$
	0.5	$\mathbf{0.039}_{\pm .045}$	$\mathbf{0.045}_{\pm .057}$	$0.050_{\pm .012}$	$\mathbf{0.013}_{\pm .004}$	$\mathbf{0.023}_{\pm .0028}$	$0.009_{\pm .0019}$
	1	$\mathbf{0.104}_{\pm .071}$	$\mathbf{0.062}_{\pm .037}$	$0.035_{\pm .013}$	$0.026_{\pm .015}$	$\mathbf{0.026}_{\pm .0003}$	$\mathbf{0.007}_{\pm .0001}$

K.1.3 INTERVENTION PERFORMANCE

Table 5: Task accuracy (%) before (Init.) and after (Final) intervening on all concept groups, mean ± s.d. over 5 folds. **Bold** denotes the significantly higher value between GC-CBM and Joint CBM at matched λ_c (two-sided Welch t-test with Holm correction, $\alpha = 0.05$)

Model	λ_c	TabularToy		AwA2		CUB-200	
		Init.	Final	Init.	Final	Init.	Final
Hard CBM	–	99.5 ± 0.3	100.0 ± 0.0	83.1 ± 0.4	99.9 ± 0.2	61.8 ± 0.3	100.0 ± 0.0
Seq CBM	–	99.6 ± 0.1	100.0 ± 0.0	83.1 ± 0.5	100.0 ± 0.0	65.9 ± 0.1	96.6 ± 0.5
Joint CBM	0.1	99.6 ± 0.2	52.4 ± 8.4	82.9 ± 0.6	86.0 ± 9.9	$\mathbf{71.0 \pm 0.8}$	14.1 ± 2.1
	0.5	$\mathbf{99.6 \pm 0.1}$	95.7 ± 5.9	83.0 ± 0.6	99.6 ± 0.2	$\mathbf{72.4 \pm 1.0}$	64.5 ± 2.9
	1.0	99.6 ± 0.1	83.9 ± 4.8	82.9 ± 0.6	99.7 ± 0.3	$\mathbf{68.0 \pm 0.6}$	89.4 ± 1.1
GC-CBM	0.1	99.1 ± 0.5	$\mathbf{100.0 \pm 0.0}$	$\mathbf{85.2 \pm 0.8}$	100.0 ± 0.0	65.9 ± 0.2	$\mathbf{96.9 \pm 0.0}$
	0.5	99.3 ± 0.2	100.0 ± 0.0	$\mathbf{85.1 \pm 0.6}$	100.0 ± 0.0	66.1 ± 0.8	$\mathbf{98.0 \pm 0.9}$
	1.0	99.3 ± 0.3	$\mathbf{100.0 \pm 0.0}$	$\mathbf{85.0 \pm 0.9}$	100.0 ± 0.0	66.5 ± 0.2	$\mathbf{98.1 \pm 0.0}$

K.2 GC-CEM RESULTS

K.2.1 TASK AND CONCEPT ACCURACY

Table 6: **Task accuracy (Task) and concept accuracy (Con) for CEM and GC-CEM.** Mean ± std over 5 folds. **Bold** denotes the significantly higher value between Joint CEM and GC-CEM at matched λ_c (two-sided Welch t-test with Holm correction at $\alpha = 0.05$)

		TabularToy		**AwA2**		**CUB**	
Model	λ_c	Task	Con	Task	Con	Task	Con
CEM	0.1	$99.6_{\pm 0.07}$	$98.7_{\pm 1.79}$	$83.0_{\pm 0.52}$	$95.7_{\pm 0.14}$	$\mathbf{72.1}_{\pm 1.12}$	$91.1_{\pm 0.49}$
	0.5	$99.6_{\pm 0.09}$	$99.8_{\pm 0.08}$	$82.6_{\pm 0.55}$	$95.7_{\pm 0.12}$	$\mathbf{73.2}_{\pm 0.68}$	$93.8_{\pm 0.25}$
	1.0	$\mathbf{99.8}_{\pm 0.13}$	$\mathbf{99.9}_{\pm 0.04}$	$82.8_{\pm 0.39}$	$95.8_{\pm 0.13}$	$\mathbf{73.9}_{\pm 0.36}$	$95.1_{\pm 0.05}$
GC-CEM	0.1	$99.5_{\pm 0.28}$	$99.7_{\pm 0.14}$	$82.2_{\pm 0.34}$	$95.7_{\pm 0.13}$	$69.1_{\pm 1.02}$	$\mathbf{95.1}_{\pm 0.21}$
	0.5	$99.5_{\pm 0.13}$	$99.7_{\pm 0.15}$	$83.5_{\pm 0.45}$	$\mathbf{96.1}_{\pm 0.14}$	$70.6_{\pm 0.44}$	$\mathbf{95.4}_{\pm 0.10}$
	1.0	$99.3_{\pm 0.13}$	$99.6_{\pm 0.05}$	$83.8_{\pm 0.85}$	$\mathbf{96.2}_{\pm 0.19}$	$70.1_{\pm 0.58}$	$\mathbf{95.3}_{\pm 0.08}$

K.2.2 RESIDUAL TASK LEAKAGE AND RESIDUAL INTERCONCEPT LEAKAGE

Table 7: **RTL** and **RCL on concept vectors** across models, datasets and λ_c values. Results are mean ± std over 5 folds. **Bold** denotes the significantly lower value between Joint CBM and GC-CBM at matched λ_c (two-sided Welch t-test with Holm correction at $\alpha = 0.05$). CUB RCL standard deviations are given to 4 d.p. as they fall below 0.001

		TabularToy		**AwA2**		**CUB**	
Model	λ_c	RTL	RCL	RTL	RCL	RTL	RCL
CEM	0.1	$0.109_{\pm 0.035}$	$0.124_{\pm 0.019}$	$0.468_{\pm 0.012}$	$0.014_{\pm 0.0007}$	$0.416_{\pm 0.022}$	$0.008_{\pm 0.0011}$
	0.5	$0.062_{\pm 0.016}$	$0.106_{\pm 0.014}$	$0.428_{\pm 0.024}$	$0.012_{\pm 0.0008}$	$0.454_{\pm 0.017}$	$\mathbf{0.008}_{\pm 0.0010}$
	1.0	$0.056_{\pm 0.010}$	$0.088_{\pm 0.018}$	$0.424_{\pm 0.024}$	$0.012_{\pm 0.0003}$	$0.469_{\pm 0.009}$	$\mathbf{0.007}_{\pm 0.0001}$
GC-CEM	0.1	$\mathbf{0.001}_{\pm 0.001}$	$\mathbf{0.002}_{\pm 0.001}$	$\mathbf{0.009}_{\pm 0.002}$	$\mathbf{0.0002}_{\pm 0.0001}$	$\mathbf{0.203}_{\pm 0.003}$	$0.009_{\pm 0.0003}$
	0.5	$\mathbf{0.001}_{\pm 0.000}$	$\mathbf{0.003}_{\pm 0.001}$	$\mathbf{0.024}_{\pm 0.005}$	$\mathbf{0.0008}_{\pm 0.0002}$	$\mathbf{0.265}_{\pm 0.005}$	$0.011_{\pm 0.0003}$
	1.0	$\mathbf{0.001}_{\pm 0.000}$	$\mathbf{0.002}_{\pm 0.001}$	$\mathbf{0.039}_{\pm 0.005}$	$\mathbf{0.0011}_{\pm 0.0002}$	$\mathbf{0.281}_{\pm 0.004}$	$0.012_{\pm 0.0002}$

K.2.3 INTERVENTION PERFORMANCE

Table 8: Task accuracy (%) before (Init.) and after (Final) intervening on all concept groups, mean $\pm$ s.d. over 5 folds. **Bold** denotes the significantly higher value between GC-CEM and CEM at matched λ_c (two-sided Welch t-test with Holm correction, $\alpha = 0.05$).

Model	λ_c	TabularToy		AwA2		CUB-200	
		Init.	Final	Init.	Final	Init.	Final
CEM	0.1	99.6 ± 0.1	99.7 ± 0.1	$\mathbf{83.1 \pm 0.4}$	83.6 ± 0.3	$\mathbf{72.1 \pm 1.1}$	74.6 ± 0.2
	0.5	99.6 ± 0.1	99.8 ± 0.1	82.7 ± 0.4	83.1 ± 0.4	$\mathbf{73.2 \pm 0.7}$	78.3 ± 0.4
	1.0	$\mathbf{99.8 \pm 0.1}$	99.7 ± 0.2	82.9 ± 0.3	82.8 ± 0.3	$\mathbf{73.9 \pm 0.4}$	81.1 ± 0.3
GC-CEM	0.1	99.5 ± 0.3	99.8 ± 0.2	82.2 ± 0.4	84.5 ± 0.8	69.1 ± 1.0	$\mathbf{85.4 \pm 0.6}$
	0.5	99.5 ± 0.1	99.9 ± 0.1	83.5 ± 0.5	$\mathbf{85.6 \pm 0.7}$	70.6 ± 0.4	$\mathbf{87.2 \pm 0.7}$
	1.0	99.3 ± 0.1	99.7 ± 0.1	83.8 ± 0.9	85.2 ± 1.6	70.1 ± 0.6	$\mathbf{87.4 \pm 0.4}$