DIGGING DEEPER: LEARNING MULTI-LEVEL CONCEPT HIERARCHIES

Anonymous authors
Paper under double-blind review

ABSTRACT

Concept-based models promise interpretability by explaining predictions with human-understandable concepts, but they typically rely on exhaustive annotations and treat concepts as flat and independent. Hierarchical Concept Embedding Models (HiCEMs) address this by modelling concept relationships, and Concept Splitting enables their use with only coarse annotations by discovering sub-concepts. However, both are restricted to shallow hierarchies. We overcome this limitation with *Deep-HiCEMs*, which model arbitrarily deep hierarchies, and *Multi-Level Concept Splitting* (MLCS), which uncovers multi-level concept hierarchies from only top-level supervision. Experiments across multiple datasets show that MLCS discovers human-interpretable concepts absent during training and that Deep-HiCEMs maintain high accuracy while supporting test-time concept interventions that can improve task performance.

1 INTRODUCTION

Concept-based models (Koh et al., 2020; Espinosa Zarlenga et al., 2022; Oikarinen et al., 2023) aim to make neural networks interpretable by predicting human-understandable concepts and using them to explain decisions. By exposing intermediate concepts, these models allow users to inspect, debug, and intervene on model reasoning. However, most concept-based approaches assume concepts are flat and independent, leading to their representations failing to capture known inter-concept relationships (Raman et al., 2024). This is problematic because real-world concepts are often interrelated, and human cognition utilises these relationships (McClelland & Rogers, 2003).

Hierarchical Concept Embedding Models (HiCEMs, Anonymous (2026)) explicitly model concept relationships, while Concept Splitting reduces annotation costs by discovering sub-concepts using only coarse top-level concept labels. However, these methods are restricted to shallow hierarchies, supporting only a single layer of sub-concepts in addition to the coarse provided concepts. This prevents models from capturing richer structure and from offering interventions at several levels of abstraction.

To address these issues, we propose *Multi-Level Concept Splitting* (MLCS), a method for discovering multi-level concept hierarchies from concept embeddings trained with only top-level supervision, and *Deep-HiCEM*, an architecture designed to support arbitrarily deep hierarchies of concepts. Together, these enable hierarchical explanations and interventions without requiring exhaustive concept annotations.

We evaluate our approach across several datasets, including a synthetic dataset designed for hierarchical concept discovery. Our results show that multi-level human-interpretable concept hierarchies can be discovered reliably, and used to explain predictions without sacrificing performance. Our contributions are:

- We introduce **MLCS** for discovering multi-level concept hierarchies from only top-level supervision.
- We propose **Deep-HiCEMs**, which model arbitrarily deep concept hierarchies and allow human interventions at any level in the hierarchy.
- We demonstrate that Deep-HiCEMs trained via MLCS can accurately discover interpretable concept hierarchies that were absent during training. Moreover, our experiments show that

Deep-HiCEMs trained with MLCS achieve competitive task accuracies and are receptive to test-time concept interventions at different levels of granularity.

2 Background and Related Work

Concept learning Concept-based methods aim to explain a model's predictions using human-understandable concepts (e.g., "colour" or "size") (Bau et al., 2017; Fong & Vedaldi, 2018; Kim et al., 2018). Some methods, such as Concept Bottleneck Models (CBMs, Koh et al. (2020)) and Concept Embedding Models (CEMs, Espinosa Zarlenga et al. (2022)), explicitly incorporate concepts in their architecture, leading to inherently interpretable models that provide concept-based explanations. These methods typically require a concept-annotated training set and may suffer from suboptimal predictive performance due to conflicting training objectives (Espinosa Zarlenga et al., 2022). Furthermore, they often do not consider the relationships between concepts; instead, they treat all concepts as independent variables (Havasi et al., 2022).

Concept Splitting and HiCEMs Concept Splitting (Anonymous, 2026) is a method for automatically discovering fine-grained sub-concepts from a model trained with only coarse concept labels. It uses sparse autoencoders (Bricken et al., 2023) to discover sub-concepts in concept-aligned embeddings taken from a pretrained CEM (Espinosa Zarlenga et al., 2022). This reveals interpretable, finer concepts without extra annotation, enabling more granular explanations and interventions. However, Concept Splitting is limited to a single additional level of granularity: it can reveal sub-concepts of a provided concept, but it cannot expose deeper hierarchical structure among those sub-concepts. In this work, we introduce MLCS to overcome this limitation.

HiCEMs (Anonymous, 2026) were proposed as an architectural counterpart to Concept Splitting, designed to represent hierarchical concept structure within an inherently interpretable model. HiCEMs organise concepts into parent–child groups. In a HiCEM, from a latent code $\mathbf{h}$, we learn two embeddings per top-level concept. The positive embedding, $\hat{\mathbf{c}}_i^{+\prime}$, represents concept c_i's active state, and the negative embedding, $\hat{\mathbf{c}}_i^{-\prime}$, represents its inactive state. These embeddings are then passed through sub-concepts modules, which produce new embeddings ($\hat{\mathbf{c}}_i^{+}$ and $\hat{\mathbf{c}}_i^{-}$) that include information about sub-concepts. The sub-concepts modules also output the most likely sub-concept probabilities, which are used to calculate top-level concept probabilities. These probabilities are used to output an embedding for each concept via a weighted mixture of positive and negative embeddings. HiCEMs support concept interventions: a user can correct concept predictions at test time, and the downstream task prediction is recomputed using the corrected concepts, enabling targeted human-in-the-loop control. Importantly, the original HiCEM formulation focuses on shallow hierarchies with only two levels (concepts and sub-concepts). This leaves open the question of how to represent and learn deeper, multi-level hierarchies, which we address with Deep-HiCEM, a generalisation that supports hierarchical concept structure beyond a single parent–child layer.

3 Multi-Level Concept Splitting

Concept Splitting (Anonymous, 2026) discovers fine-grained concepts by training Sparse AutoEncoders (SAEs, Bricken et al. (2023)) on a pretrained CEM's (Espinosa Zarlenga et al., 2022) concept embeddings and turning sparse features into new concept labels. However, it is limited to a single additional level of granularity: it can reveal sub-concepts of a provided concept, but it cannot expose deeper hierarchical structure among those sub-concepts.

We address this limitation with MLCS, which replaces the single-level SAE with a Hierarchical Sparse AutoEncoder (HiSAE) that learns structured sparse features at multiple levels. This enables the discovery of both sub-concepts and sub-sub-concepts from the same embedding space, allowing us to construct deeper concept hierarchies from only top-level concept supervision.

The use of a HiSAE is the primary difference between MLCS and Concept Splitting, so we focus here on the HiSAE architecture. The HiSAE is designed to learn sparse structure at two or more levels simultaneously: a top level captures candidate sub-concepts (e.g., apple), and a lower level conditioned on each discovered sub-concept captures finer-grained refinements (e.g., red apple).

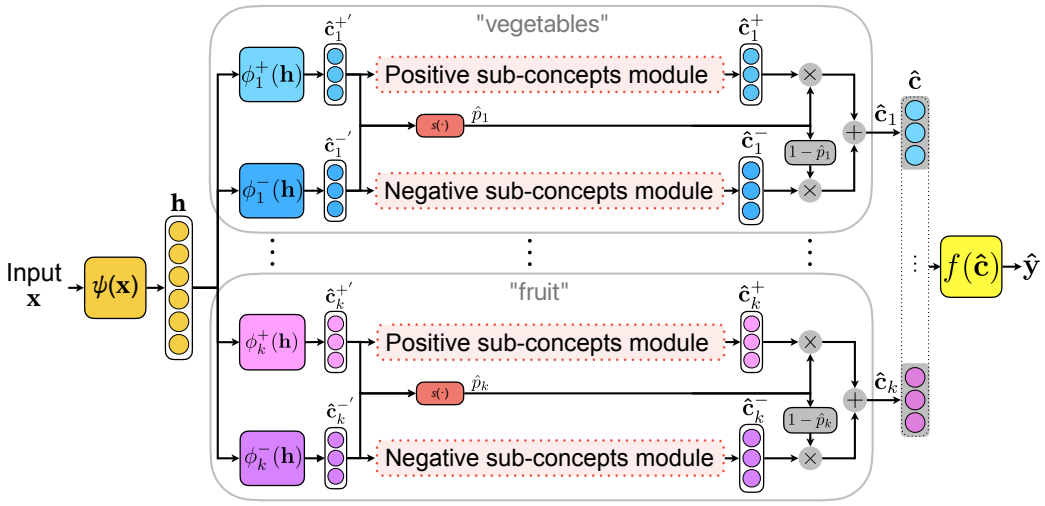

Figure 1: **Deep-HiCEM:** as in a HiCEM, from a latent code $\mathbf{h}$, we learn two embeddings per concept ($\hat{\mathbf{c}}_\mathbf{i}^{+\prime}$ and $\hat{\mathbf{c}}_\mathbf{i}^{-\prime}$), which are then passed through sub-concepts modules, which produce new embeddings ($\hat{\mathbf{c}}_\mathbf{i}^{+}$ and $\hat{\mathbf{c}}_\mathbf{i}^{-}$) that include information about sub-concepts and their descendants.

3.1 Hierarchical Sparse Autoencoder

The HiSAE, inspired by Muchane et al. (2025), consists of (i) a top-level encoder that maps an input embedding to a dictionary of size K, (ii) a set of sub-encoders, one per top-level latent, each mapping the input to a sub-dictionary of size K_s, and (iii) corresponding linear decoders for both levels.

Given an input embedding $\mathbf{e}$, the top-level encoder produces activations over the K top-level latents. A top-k sparsification step (Bussmann et al., 2024) retains only the k largest activations.

For each of the k *active* top-level latents ℓ, the corresponding sub-encoder processes the input embedding $\mathbf{e}$ to produce activations over its sub-dictionary. A second top-k_s operation selects k_s active sub-latents. The reconstruction is formed by summing the contributions of the active top-level and sub-level decoders, and the model is trained with a standard mean squared error reconstruction loss.

Crucially, the sub-level is gated by the top-level: sub-latents only contribute when their parent latent is active. This ties sub-sub-concepts to their parent sub-concepts.

3.2 Depth of the Hierarchy

In principle, the HiSAE architecture can be extended recursively, yielding arbitrarily deep hierarchies by attaching further sparse autoencoding stages to sub-latents. MLCS therefore provides a general mechanism for hierarchical concept discovery from embedding spaces.

In this work, however, we restrict ourselves to *two discovered levels*: sub-concepts and sub-sub-concepts. Even with this restriction, MLCS allows models to represent richer conceptual structure than single-level Concept Splitting while still requiring supervision only at the top level.

4 Deep Hierarchical CEMs

We introduce the Deep-HiCEM architecture (Figure 1), which extends the HiCEM architecture to support deeper concept hierarchies, like those produced by MLCS. We focus on the differences from HiCEMs; we do not discuss training Deep-HiCEMs in detail as the process is the same as training HiCEMs.

4.1 Concept Structure

Deep-HiCEMs represent concepts that have been organised into a tree. Each node in the tree represents a concept and can have both *positive* and *negative* sub-concepts. Positive sub-concepts can only be present in an example if their parent concept is present, and negative sub-concepts can only be present if their parent concept is not. The concept trees may be arbitrarily deep.

4.2 Architecture

As in HiCEMs, for each top-level concept, a Deep-HiCEM learns a mixture of two embeddings with semantics representing the concept's activity. Each top-level concept c_i is represented with the embeddings $\hat{\mathbf{c}}_i^+, \hat{\mathbf{c}}_i^- \in \mathbb{R}^m$. Here, $\hat{\mathbf{c}}_i^+$ represents c_i's active state, and $\hat{\mathbf{c}}_i^-$ represents its inactive state. We also want $\hat{\mathbf{c}}_i^+$ and $\hat{\mathbf{c}}_i^-$ to contain information about c_i's positive and negative sub-concepts (and their descendants), respectively. To achieve this, a backbone network $\psi(\mathbf{x})$ produces a latent representation $\mathbf{h} \in \mathbb{R}^{n_{\text{hidden}}}$ which is the input to top-level embedding generators ϕ_i^+ and ϕ_i^-. These top-level embedding generators produce intermediate embeddings $\hat{\mathbf{c}}_i^{+\prime} = \phi_i^+(\mathbf{h}), \hat{\mathbf{c}}_i^{-\prime} = \phi_i^-(\mathbf{h}) \in \mathbb{R}^m$. As in HiCEMs, we implement the top-level embedding generators as single fully connected layers.

To produce final concept embeddings that contain information about sub-concepts, the embeddings $\hat{\mathbf{c}}_i^{+\prime}$ and $\hat{\mathbf{c}}_i^{-\prime}$ are passed through a positive and a negative *sub-concepts module*, respectively. The positive sub-concepts module, which we describe in further detail below, is responsible for learning the positive sub-concepts of c_i, and their descendants. It outputs the positive concept embedding for c_i, $\hat{\mathbf{c}}_i^+$. Similarly, the negative sub-concepts module outputs the negative concept embedding for c_i, $\hat{\mathbf{c}}_i^-$. If concept c_i has no positive sub-concepts, then we take $\hat{\mathbf{c}}_i^+ = \hat{\mathbf{c}}_i^{+\prime}$. We proceed analogously in the absence of negative sub-concepts. The probability of concept c_i is calculated as $\hat{p}_i = s([\hat{\mathbf{c}}_i^{+'}, \hat{\mathbf{c}}_i^{-'}]^T)$, where s is a shared scoring function that calculates concept probabilities from concept embeddings. As in HiCEMs, the final concept embedding $\hat{\mathbf{c}}_i$ for c_i is calculated as a weighted mixture of $\hat{\mathbf{c}}_i^+$ and $\hat{\mathbf{c}}_i^-$: $\hat{\mathbf{c}}_i = \hat{p}_i \hat{\mathbf{c}}_i^+ + (1 - \hat{p}_i)\hat{\mathbf{c}}_i^-$. Downstream task predictions are calculated as in HiCEMs.

Like HiCEMs, Deep-HiCEMs support concept interventions, and when a sub-concept is intervened on this may also result in its parent being updated (e.g., if a human expert informs the model that a positive sub-concept is present, this implies its parent is also present).

4.3 Sub-concepts Modules

We describe a positive sub-concepts module, but negative sub-concepts modules operate in exactly the same way. Sub-concept embeddings are produced in the same way as top-level concept embeddings (Figure 1, Section 4.2), with nested sub-concepts modules if the sub-concept has sub-concepts itself. However, instead of operating on the latent code $\mathbf{h}$, sub-concept embedding generators take as input the preliminary embedding of the parent concept. The concatenated mixed embeddings for all the sub-concepts are passed through an embedding compressor (implemented as a single fully connected layer) to produce a fixed-length positive embedding for the parent concept, $\hat{\mathbf{c}}_i^+$.

5 Experiments

We evaluate MLCS and Deep-HiCEMs by exploring the following research questions:

RQ1 Does MLCS discover interpretable concept hierarchies?

RQ2 How do Deep-HiCEMs' task accuracies compare to those of standard HiCEMs?

RQ3 Can a Deep-HiCEM's task accuracy be improved by intervening on discovered concepts?

5.1 PseudoKitchens-2

To evaluate the ability of MLCS and Deep-HiCEMs to discover and represent two levels of hierarchy (sub-concepts and sub-sub-concepts), we adapt the PseudoKitchens dataset introduced by Anonymous (2026) so that both sub-concepts and sub-sub-concepts can be discovered. The discoverable sub-concepts are ingredients (e.g., apple), and the sub-sub-concepts correspond to different variants of that ingredient (e.g., red apple). Full details are given in Appendix A.

Table 1: Mean ROC-AUC for discovered concepts. Concepts discovered with MLCS are predicted accurately.

	MNIST-ADD	SHAPES	CUB	AwA2	PseudoKitchens-2
LF-CBM	–	$0.75_{\pm 0.00}$	$0.77_{\pm 0.00}$	$0.78_{\pm 0.00}$	–
HiCEM + Concept Splitting	$0.93_{\pm 0.01}$	$\mathbf{0.93_{\pm 0.01}}$	$\mathbf{0.85_{\pm 0.01}}$	$\mathbf{0.88_{\pm 0.01}}$	$\mathbf{0.80_{\pm 0.00}}$
Deep-HiCEM + MLCS (ours)	$\mathbf{0.94_{\pm 0.01}}$	$\mathbf{0.93_{\pm 0.01}}$	$0.84_{\pm 0.01}$	$0.85_{\pm 0.01}$	$0.79_{\pm 0.01}$

5.2 Setup

Datasets We evaluate our method across five datasets: MNIST-ADD (Anonymous (2026), based on LeCun et al. (2010)), the SHAPES dataset (Anonymous, 2026), Caltech-UCSD Birds-200-2011 (CUB) (Wah et al., 2011), Animals with Attributes 2 (AwA2) (Xian et al., 2019), and our PseudoKitchens-2 dataset. Full details on PseudoKitchens-2 are included in Appendix A, and details for the other datasets are given by Anonymous (2026).

Metrics Following Anonymous (2026), we first run MLCS on the provided concepts using an initial CEM, and then train a Deep-HiCEM with both the provided top-level concepts and the concepts discovered by MLCS. We evaluate discovered concepts and perform interventions by automatically aligning discovered concepts with a human-interpretable "left-out" concept from a predefined "concept bank". This bank contains anticipated concepts that are excluded during the initial CEM training (e.g., "the first digit is 5" in MNIST-ADD), and each bank concept is associated in advance with the parent concept whose sub-concept it may represent.

To match discovered concepts to the bank, we compute ROC-AUC scores between the discovered concept labels and their potential parent-concept-associated matches in the bank. Each concept in the bank is assigned to the sub-concept with the highest ROC-AUC score, as long as that score is greater than 0.7. Discovered concepts without a bank match above this threshold are excluded from evaluation.

For PseudoKitchens-2, the bank contains both sub-concepts (e.g., "apple") and sub-sub-concepts (e.g., "red apple"). When matching discovered sub-concepts to the concept bank, we use the average ROC-AUC of the sub-concept and any associated sub-sub-concepts. For all other datasets, we match and evaluate only sub-concepts, as ground truth sub-sub-concept labels are not available.

To answer **RQ1**, we report the average discovered concept ROC-AUC of the Deep-HiCEM. For **RQ2**, we compare task accuracy and provided-concept ROC-AUC between a standard HiCEM and a Deep-HiCEM. For **RQ3**, we measure the change in the task accuracy of Deep-HiCEMs as concepts are intervened. All metrics are computed on the test sets using three random seeds, and we report means and standard deviations. Concept accuracy is always summarised using mean ROC-AUC to avoid misleading results from majority-class predictors. For details on model architectures, training and hyperparameters see Appendix B.

Baselines We compare our Deep-HiCEMs against standard HiCEMs (Anonymous, 2026). We also report results for the baselines considered by Anonymous (2026): black-box models, CEMs (Espinosa Zarlenga et al., 2022), CBMs (Koh et al., 2020), label-free CBMs (*LF-CBMs*, (Oikarinen et al., 2023)), Post-hoc CBMs (*PCBMs*, (Yuksekgonul et al., 2023)), and PCBMs with residual connections (*PCBM-hs*). For the PseudoKitchens-2 dataset, we implement the baselines following the protocol of Anonymous (2026); for the other datasets, we use the results reported in their work.

5.3 Results

Discovered concepts are human-interpretable and can be predicted accurately (RQ1, Table 1). We report the accuracy of the discovered concept predictions made by our models using the ground truth labels of the corresponding human-interpretable concept bank concepts on the test datasets. Table 1 shows that the mean discovered concept ROC-AUCs of Deep-HiCEMs are consistently high, always within a few percentage points of those of HiCEMs. This indicates that the labels produced by MLCS align Deep-HiCEM concept activations with human-interpretable concepts. On PseudoKitchens-2, the majority of sub-sub-concepts in the bank were successfully matched to

Table 2: Task accuracies. The task accuracy of Deep-HiCEMs is competitive with all our baselines.

	MNIST-ADD	SHAPES	CUB	AwA2	PseudoKitchens-2
Black box (not interpretable)	$0.94_{\pm 0.00}$	$0.89_{\pm 0.00}$	$0.80_{\pm 0.00}$	$0.98_{\pm 0.00}$	$0.63_{\pm 0.02}$
LF-CBM	–	$0.59_{\pm 0.01}$	$\mathbf{0.80_{\pm 0.00}}$	$0.94_{\pm 0.00}$	–
PCBM	$0.16_{\pm 0.03}$	$0.54_{\pm 0.01}$	$0.65_{\pm 0.01}$	$0.95_{\pm 0.00}$	$0.12_{\pm 0.00}$
PCBM-h	$0.53_{\pm 0.01}$	$0.73_{\pm 0.00}$	$0.73_{\pm 0.00}$	$0.96_{\pm 0.00}$	$0.48_{\pm 0.00}$
CBM	$0.23_{\pm 0.01}$	$0.78_{\pm 0.01}$	$0.65_{\pm 0.00}$	$0.97_{\pm 0.00}$	$\mathbf{0.60_{\pm 0.01}}$
CEM	$\mathbf{0.92_{\pm 0.01}}$	$\mathbf{0.89_{\pm 0.00}}$	$0.76_{\pm 0.01}$	$\mathbf{0.98_{\pm 0.00}}$	$0.59_{\pm 0.01}$
HiCEM + Concept Splitting	$\mathbf{0.92_{\pm 0.00}}$	$0.87_{\pm 0.02}$	$0.74_{\pm 0.01}$	$\mathbf{0.98_{\pm 0.00}}$	$0.57_{\pm 0.03}$
Deep-HiCEM + MLCS (ours)	$\mathbf{0.92_{\pm 0.00}}$	$0.87_{\pm 0.01}$	$0.73_{\pm 0.01}$	$0.97_{\pm 0.00}$	$0.58_{\pm 0.01}$

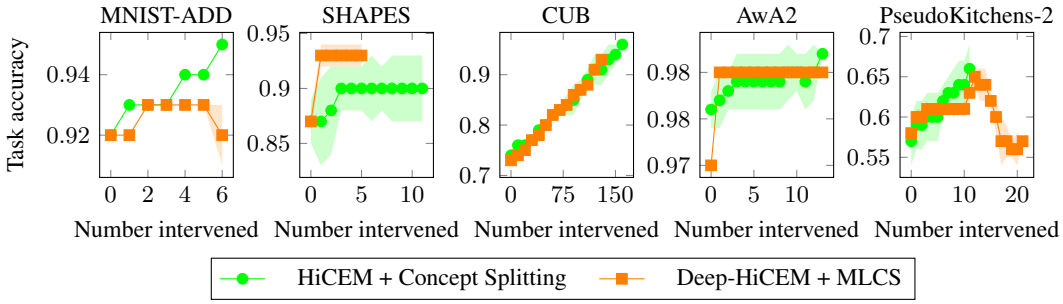

Figure 2: Task accuracy as discovered concepts are intervened. Intervening on discovered concepts improves task accuracy, with a few exceptions that would benefit from further investigation.

discovered sub-sub-concepts on all runs, demonstrating that MLCS and Deep-HiCEMs can discover and represent deeper concept hierarchies while maintaining discovered-concept ROC-AUCs close to those of HiCEMs after Concept Splitting.

Deep-HiCEMs have high task accuracy (RQ2, Table 2). We measure the task accuracy of Deep-HiCEMs and our baselines. The results are in Table 2. Deep-HiCEMs achieve high task accuracy compared to the baselines. In particular, the task accuracy of Deep-HiCEMs is never more than 1% below that of standard HiCEMs, so the discovery of deeper hierarchies does not lead to a reduction in task accuracy. We investigate the provided concept accuracy of Deep-HiCEMs in Appendix C, and find that Deep-HiCEMs can predict provided concepts as well as HiCEMs.

Intervening on concepts discovered by MLCS can enhance task accuracy, with a few exceptions that would benefit from further investigation (RQ3, Figure 2) As shown in Figure 2, intervening on discovered concepts can lead to an increase in task accuracy, although interventions on some discovered concepts have no effect or decrease task accuracy. The decreases in task accuracy caused by discovered concept interventions, particularly in PseudoKitchens-2, would benefit from further investigation, and we suggest this for future work. We evaluate interventions on the provided top-level concepts in Appendix D and find that provided concept interventions perform equally well in Deep-HiCEMs and HiCEMs. Overall, these results indicate that many of the concepts uncovered by MLCS are not only interpretable but also actionable, and that leveraging them through interventions can translate into performance gains.

6 Limitations and Conclusion

We introduced MLCS and Deep-HiCEMs, a framework for discovering and modelling deep concept hierarchies from only top-level supervision. Across several datasets, we showed that our approach reliably uncovers human-interpretable concept hierarchies absent during training, while maintaining competitive task accuracy. However, interventions on discovered concepts are not always beneficial, and in some cases can slightly reduce task accuracy. Despite this, many discovered concepts are both interpretable and respond well to interventions, and modelling deeper concept structure is a practical step towards more expressive concept-based interpretability.

REFERENCES

Anonymous. Hierarchical concept-based interpretable models. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=h61OIERd38.

David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3319–3327, Los Alamitos, CA, USA, 07 2017. IEEE Computer Society. doi: 10.1109/CVPR.2017.354. URL https://doi.ieeecomputersociety.org/10.1109/CVPR.2017.354.

Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Zac Hatfield-Dodds, Alex Tamkin, Karina Nguyen, Brayden McLean, Josiah E Burke, Tristan Hume, Shan Carter, Tom Henighan, and Christopher Olah. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023. https://transformer-circuits.pub/2023/monosemantic-features/index.html.

Bart Bussmann, Patrick Leask, and Neel Nanda. Batchtopk sparse autoencoders. In *NeurIPS 2024 Workshop on Scientific Methods for Understanding Deep Learning*, 2024. URL https://openreview.net/forum?id=d4dpOCqybL.

Mateo Espinosa Zarlenga, Pietro Barbiero, Gabriele Ciravegna, Giuseppe Marra, Francesco Giannini, Michelangelo Diligenti, Zohreh Shams, Frederic Precioso, Stefano Melacci, Adrian Weller, Pietro Lió, and Mateja Jamnik. Concept embedding models: Beyond the accuracy-explainability trade-off. In *Advances in Neural Information Processing Systems*, volume 35, pp. 21400–21413. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/867c06823281e506e8059f5c13a57f75-Paper-Conference.pdf.

Ruth Fong and Andrea Vedaldi. Net2vec: Quantifying and explaining how concepts are encoded by filters in deep neural networks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8730–8738, Los Alamitos, CA, USA, 06 2018. IEEE Computer Society. doi: 10.1109/CVPR.2018.00910. URL https://doi.ieeecomputersociety.org/10.1109/CVPR.2018.00910.

Marton Havasi, Sonali Parbhoo, and Finale Doshi-Velez. Addressing Leakage in Concept Bottleneck Models. In *Advances in Neural Information Processing Systems*, 2022.

Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, and Rory Sayres. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (TCAV). In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2668–2677. PMLR, 07 2018. URL https://proceedings.mlr.press/v80/kim18d.html.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://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. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5338–5348. PMLR, 07 2020. URL https://proceedings.mlr.press/v119/koh20a.html.

Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. Online, 2010. URL http://yann.lecun.com/exdb/mnist.

James L. McClelland and Timothy T. Rogers. The parallel distributed processing approach to semantic cognition. *Nature Reviews Neuroscience*, 4(4):310–322, 04 2003. ISSN 1471-0048. doi: 10.1038/nrn1076. URL https://doi.org/10.1038/nrn1076.

Mark Muchane, Sean Richardson, Kiho Park, and Victor Veitch. Incorporating hierarchical semantics in sparse autoencoder architectures, 2025. URL https://arxiv.org/abs/2506.01197.

Tuomas P. Oikarinen, Subhro Das, Lam M. Nguyen, and Tsui-Wei Weng. Label-free concept bottleneck models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/pdf?id=FlCg47MNvBA.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. *PyTorch: an imperative style, high-performance deep learning library*. Curran Associates Inc., Red Hook, NY, USA, 2019.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in python. *J. Mach. Learn. Res.*, 12(null):2825–2830, November 2011. ISSN 1532-4435.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8748–8763. PMLR, 2021. URL http://proceedings.mlr.press/v139/radford21a.html.

Naveen Raman, Mateo Espinosa Zarlenga, and Mateja Jamnik. Understanding inter-concept relationships in concept-based models. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. Technical Report CNS-TR-2011-001, August 2011.

Yongqin Xian, Christoph H. Lampert, Bernt Schiele, and Zeynep Akata. Zero-shot learning—a comprehensive evaluation of the good, the bad and the ugly. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(9):2251–2265, 2019. doi: 10.1109/TPAMI.2018.2857768.

Mert Yuksekgonul, Maggie Wang, and James Zou. Post-hoc concept bottleneck models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=nA5AZ8CEyow.

A PSEUDOKITCHENS-2

This appendix describes PseudoKitchens-2, an extension of the PseudoKitchens dataset (Anonymous, 2026) designed to enable the discovery and evaluation of two-level concept hierarchies.

PseudoKitchens-2 is generated in the same way as PseudoKitchens (Anonymous, 2026). The recipes described by Anonymous (2026) are altered so that different variations of the same ingredient are distinguished (e.g., some recipes require "red apples" while others require "green apples"). The ingredients that have variations are apple (5 variations), potato (5 variations), and pepper (3 variations).

A.1 RECIPES

We adapted the recipes used by Anonymous (2026) that define valid combinations of ingredients for the classification task. Some ingredients are organised into groups as shown in Table 3 (these groups are the same as the ones used by Anonymous (2026)). If a recipe contains an ingredient group, a random number of ingredients are selected from that group, unless the group is pasta in which case only one type of pasta is selected. The recipes used in PseudoKitchens-2 are shown in Table 4. Where a list of numbers is specified in brackets after an ingredient (e.g., "Apple (1)"), this refers to the variants of that ingredient that are acceptable. For each instance, a recipe is chosen uniformly at random.

Table 3: Ingredient groups in PseudoKitchens and PseudoKitchens-2.

Group	Ingredients
Fruit	Banana, Orange, Apple, Pear, Pineapple
Vegetables	Onion, Carrot, Potato, Pepper, Courgette
Pasta	Macaroni, Spaghetti

Table 4: Recipes in PseudoKitchens-2.

Recipe	Ingredients
Fruit Salad	Fruit
Vegetable Pasta	Pasta, Onion, Garlic, Oil, Vegetables, Spice, Tin Tomatoes
Risotto	Cheese, Onion, Garlic, Vegetables, Oil, Spice, Rice
Chips	Potato (2, 3, 4), Oil, Flour, Garlic, Spice
Chilli	Mince, Oil, Onion, Garlic, Chilli, Tin Tomatoes, Spice, Rice
Smoothie	Milk, Yoghurt, Fruit
Hot Chocolate	Chocolate, Milk
Banana Bread	Butter, Sugar, Egg, Flour, Banana
Chocolate Fudge Cake	Egg, Sugar, Oil, Flour, Chocolate, Syrup, Milk
Carbonara	Garlic, Meat, Butter, Cheese, Egg, Spaghetti, Spice
Apple Crumble	Apple (3), Sugar, Flour, Butter
Salad	Pepper (2, 3), Apple (1, 2, 4, 5), Potato (1, 5)

The concepts provided to the initial CEM are the ingredient groups in Table 3 (e.g., "contains fruit"), as well as all the ingredients that are not part of a group. The concept bank sub-concepts correspond to the ingredients in the ingredient groups (e.g., "contains apples"), and the sub-sub-concepts correspond to variations of the same ingredient (e.g., "contains green apples").

A.2 DATASET COMPOSITION

Like the original PseudoKitchens, the complete PseudoKitchens-2 dataset comprises 10,000 training images, 1,000 validation images, and 1,000 test images. Each image is rendered at 512×512 resolution.

Table 5: Mean ROC-AUCs for provided concepts. Deep-HiCEMs are able to predict provided concepts just as well as HiCEMs.

	MNIST-ADD	SHAPES	CUB	AwA2	PseudoKitchens-2
CBM	$\mathbf{0.99}_{\pm 0.00}$	$\mathbf{1.00}_{\pm 0.00}$	$0.89_{\pm 0.00}$	$\mathbf{1.00}_{\pm 0.00}$	$0.90_{\pm 0.00}$
CEM	$\mathbf{0.99}_{\pm 0.00}$	$\mathbf{1.00}_{\pm 0.00}$	$\mathbf{0.95}_{\pm 0.00}$	$\mathbf{1.00}_{\pm 0.00}$	$\mathbf{0.91}_{\pm 0.00}$
HiCEM + Concept Splitting	$\mathbf{0.99}_{\pm 0.00}$	$\mathbf{1.00}_{\pm 0.00}$	$0.93_{\pm 0.01}$	$\mathbf{1.00}_{\pm 0.00}$	$\mathbf{0.91}_{\pm 0.00}$
Deep-HiCEM + MLCS (ours)	$\mathbf{0.99}_{\pm 0.00}$	$\mathbf{1.00}_{\pm 0.00}$	$0.94_{\pm 0.00}$	$\mathbf{1.00}_{\pm 0.00}$	$\mathbf{0.91}_{\pm 0.00}$

B Model architectures, training and hyperparameters

Model architectures We use the CLIP ViT-L/14 foundation model (Radford et al., 2021) as the backbone for all of our models and baselines. We do not fine-tune the foundation model: we just use the representations it outputs. For all the models, we precompute the representations with standard image preprocessing pipelines and do not use any data augmentations.

When training BatchTopK SAEs, like Anonymous (2026), we use the default hyperparameters in the code released by Bussmann et al. (2024) for all datasets.[1] The key distinction of this method is its enforcement of sparsity: it selects the top $n \cdot k$ activations across an entire batch of n samples. This allows the number of active features to vary per sample, targeting an average of $k = 32$ active features. The SAEs are trained for 300 epochs with a dictionary size of 12,288 and a learning rate of 3×10^{-4}.

For MLCS, we train Hierarchical Sparse Autoencoders (HiSAEs) to discover concept hierarchies. The HiSAEs use a top-level dictionary of size 4096 with top-k sparsification ($k = 32$), and for each top-level latent we learn a sub-dictionary of size 512 with top-k_s sparsification ($k_s = 16$). HiSAEs are trained with a batch size of 1000 for up to 100 epochs.

All of our CEM and HiCEM concept embeddings have $m = 16$ activations. Across all datasets we always use a single fully connected layer for label predictor f.

Training hyperparameters Our models are trained using the Adam optimisation algorithm (Kingma & Ba, 2015) with a learning rate of 1×10^{-3}. They are trained for a maximum of 300 epochs, and training is stopped if the validation loss does not improve for 75 epochs. We use a batch size of 256.

In all CEMs, (Deep-)HiCEMs and CBMs the weight of the concept loss is set to $\lambda = 10$. Following (Koh et al., 2020), in MNIST-ADD, CUB and PseudoKitchens-2 we use a weighted cross entropy loss for concept prediction to mitigate imbalances in concept labels. In MNIST-ADD, we also use a weighted cross entropy loss for task prediction to mitigate imbalances in task labels.

When training CEMs and (Deep-)HiCEMs, the RandInt (Espinosa Zarlenga et al., 2022) regularisation strategy is used: at training time, concepts are intervened independently at random, with the probability of an intervention being $p_{\text{int}} = 0.25$. We choose $p_{\text{int}} = 0.25$ because Espinosa Zarlenga et al. (2022) find that it enables effective interventions while giving good performance.

C Evaluating Provided Concept Accuracy

As shown in Table 5, Deep-HiCEMs are able to predict provided concepts just as well as HiCEMs.

D Evaluating Provided Concept Interventions

Figure 3 shows that provided concept interventions perform equally well in Deep-HiCEMs and HiCEMs.

[1] https://github.com/bartbussmann/BatchTopK/blob/main/config.py

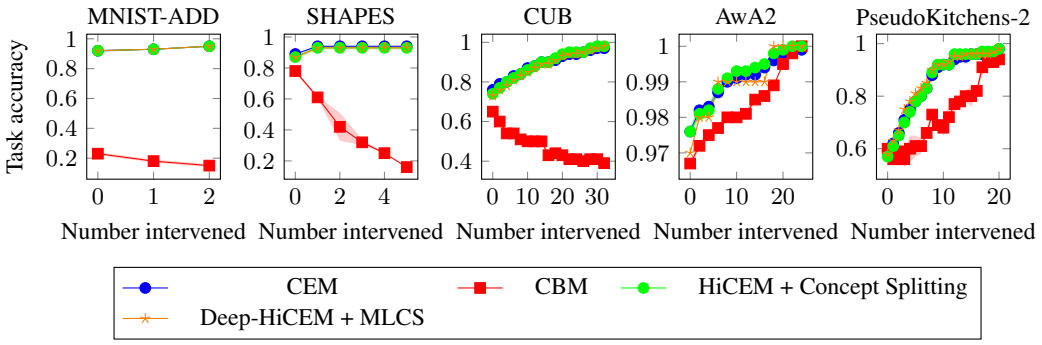

Figure 3: Change in task accuracy as provided concepts are intervened. Provided concept interventions work just as well in Deep-HiCEMs as they do in HiCEMs.

E CODE, LICENSES, AND RESOURCES

Assets We used the CLIP foundation models (Radford et al., 2021) (`https://github.com/openai/CLIP`), whose code is available under the MIT license. To run our experiments, we made use of the CEM (Espinosa Zarlenga et al., 2022) (`https://github.com/mateoespinosa/cem`, MIT license), Post-hoc CBM (Yuksekgonul et al., 2023) (`https://github.com/mertyg/post-hoc-cbm`, MIT license) and Label-free CBM (Oikarinen et al., 2023) (`https://github.com/Trustworthy-ML-Lab/Label-free-CBM`) repositories. We implemented our experiments in Python 3.12 and used open-source libraries such as PyTorch 2.9 (Paszke et al., 2019) (BSD license) and Scikit-learn (Pedregosa et al., 2011) (BSD license). We have released the code required to recreate our experiments in a MIT-licensed public repository.

Resources All of our experiments were run on virtual machines with at least 8 CPU cores, 18GB of RAM, and an NVIDIA GPU (Quadro RTX 8000 or GeForce RTX 4090). We estimate that approximately 100 GPU hours were required to complete our work.

Use of AI We used Large Language Models (LLMs) as assistants for drafting and improving the clarity and grammar of this manuscript. LLMs were also used to generate boilerplate code. However, all core research ideas, experimental design, and analysis of the results were conducted by the authors.