Train it, Extend it, Retrain it: Iterative Concept Discovery

Oscar Hill[1,*], Mateo Espinosa Zarlenga[1] and Mateja Jamnik[1]

[1]University of Cambridge, Department of Computer Science and Technology, Cambridge, CB3 0FD, United Kingdom

Abstract

Modern Deep Neural Networks are opaque: their complex latent space makes it difficult to explain their decisions. Concept Embedding Models (CEMs) aim to circumvent this limitation by predicting a set of embeddings aligned with human-understandable concepts, and then using the absence or presence of these concepts to predict a downstream task. A significant drawback of CEMs is that they require a large set of concept annotations at train-time. We relax the number of concept annotations required when training CEMs by exploiting a pre-trained CEM's learned embedding space to discover previously unknown concepts. Our method, called *Concept Excavation*, discovers new concepts by clustering concept embedding vectors from an initial CEM, and uses them to train new iterations of the model. This leads to more interpretable models that provide more complete explanations without compromising predictive accuracy. We evaluate our method on multiple datasets and show that it is able to discover concepts not provided at train time across several setups.

Keywords

Concept-based Explainability, Concept Discovery, Concept Embedding Models

1. Introduction

State-of-the-art deep neural networks (DNNs) achieve high task accuracy but cannot explain their reasoning in human-understandable terms [1]. Concept Embedding Models (CEMs) [2] address this by learning to predict a set of human-understandable *concepts* (e.g., "size" or "colour") provided at train-time, and then learning to predict a downstream task from the concepts. The concept predictions serve as an explanation for the downstream task prediction. A crucial limitation of CEMs is that they require a large and potentially expensive set of concept annotations at train-time to learn their embeddings [2].

In this paper, we show that CEMs capture concepts not provided at train-time as part of their learned embedding spaces. We propose a method, *Concept Excavation*, that exploits this by automatically extracting new concepts from the embedding space of an initial CEM, and training new iterations of the model that explicitly include these concepts. Concept Excavation works by clustering embedding vectors calculated from the examples in the training dataset, and using the clusterings to create new concept labels. The concept labels are then used to train

HI-AI@KDD, Human-Interpretable AI Workshop at the KDD 2024, 26th of August 2024, Barcelona, Spain

*Corresponding author.

ogh22@cam.ac.uk (O. Hill); me466@cam.ac.uk (M. E. Zarlenga); mateja.jamnik@cl.cam.ac.uk (M. Jamnik)

https://github.com/OscarPi/ (O. Hill); https://www.cst.cam.ac.uk/people/me466/ (M. E. Zarlenga); https://www.cl.cam.ac.uk/~mj201/ (M. Jamnik)

0009-0003-9395-1955 (O. Hill); 0009-0006-7333-5727 (M. E. Zarlenga); 0000-0003-2772-2532 (M. Jamnik)

a new CEM. The process of discovering a concept and training a model with the new concept can be repeated multiple times. Concept excavation reduces the need for large numbers of concept annotations at train-time. Using concept excavation, only a few concept annotations are needed in the training dataset, and further concepts can be discovered automatically.

We evaluate Concept Excavation by assigning human-interpretable semantics to the discovered concepts and using these to measure how accurately discovered concepts can be predicted. We perform this evaluation while also measuring the task accuracy of the CEMs after introducing discovered concepts. We summarise our contributions as follows:

- We introduce Concept Excavation, a method for training CEMs whose explanations incorporate both concepts provided at train-time as well as previously unknown discovered concepts. Our method lowers the barriers to deploying CEMs in real-world setups by reducing the need for large sets of training concept annotations and increasing the granularity of the model's explanations.
- We show that Concept Excavation discovers interpretable concepts that can be predicted with high accuracy. Additionally, CEMs trained via Concept Excavation have competitive task accuracies and are still receptive to test-time concept interventions.

2. Background

Concept Learning Concept-based methods aim to explain the predictions of AI models using human-understandable concepts [3, 4, 5, 6, 7, 8, 9]. Some methods, for example, Concept Bottleneck Models (CBMs) [10], 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 [2, 10]. Other methods attempt to construct post-hoc concept-based explanations, instead of producing inherently interpretable models. For example, Automatic Concept-based Explanations (ACE) [6] clusters a DNN's latent space to discover relevant concepts. Like ACE, we discover concepts using clustering, however we use the discovered concepts in inherently interpretable models.

Concept Embedding Models Concept Embedding Models (CEMs) [2] provide concept-based explanations while achieving higher task accuracies than CBMs. CEMs improve task accuracy by representing concepts by supervised vectors. For each concept, a CEM learns two embeddings: one for when it is active (a "positive" embedding), and another for when it is inactive (a "negative" embedding). Each concept embedding is aligned to its corresponding ground truth concept through a scoring function, which learns to assign activation probabilities for each concept. These probabilities are used to output an embedding for each concept via a weighted mixture of the positive and negative embedding. The mixed concept embeddings are concatenated and passed to the label predictor.

CEMs support *concept interventions*: at test-time, an expert can correct mispredicted concepts and allow the model to update its downstream label prediction based on these corrections. Interventions can be performed by fixing a concept's embedding to the embedding that is semantically aligned with the ground truth concept label.

Figure 1: Concept Excavation. (a) Train a CEM and calculate concept embeddings. (b) Decide which set of embeddings should be clustered. (c) Cluster the embeddings. t-SNE [11] is used for visualisation. Red points represent embeddings where the concept was predicted to be present, while blue points represent embeddings where it was predicted to be absent. (d) Create concept labels. The yellow points are marked as having the new concept, the purple points are marked as unsure and the teal points are marked as not having the new concept.

A major limitation of CEMs is the need for large numbers of concept labels in their training datasets. Our work addresses this issue by exploiting a pretrained CEM's embedding space to discover concepts that were not provided during training.

3. Concept Excavation

Previous work has noted that Euclidean distance in DNN activation spaces can be an effective similarity metric [12], and used this to discover concepts [6]. We apply this to concept embeddings in CEMs. Our approach, which we name *Concept Excavation*, takes a trained CEM and its training dataset as input and proceeds as follows: first, we run the CEM on all the training examples, storing the resulting concept embedding vectors (Figure 1(a)). Each concept embedding vector is a weighted mixture of a positive embedding and a negative embedding. We partition the vectors using predictions, because the predictions tell us whether the dominant component in the mixture is a positive embedding or a negative embedding. Let E_c be the set of embedding vectors for concept c and let $E_c^{\textbf{true}}$ and $E_c^{\textbf{false}}$ be the subsets of embeddings in E_c where c was predicted to be present or absent, respectively.

For each concept c in the current CEM, we cluster $E_c^{\textbf{true}}$ and $E_c^{\textbf{false}}$ separately. We do not cluster the embeddings in $E_c^{\textbf{true}}$ and $E_c^{\textbf{false}}$ together, because in a concept's embedding space, examples with and without the concept will be separated (due to the way CEMs construct the embeddings as a weighted mixture of positive and negative embeddings). There is, therefore, a high chance we would "discover" the existing concept. We use K-Means clustering, and the number of clusters is a hyperparameter. We tried several different clustering algorithms and found that K-Means worked best in practice. The clustering with the highest silhouette score [13] is the one used to discover a concept (Figure 1(b)). Small embedding sets are not clustered because the clusters need to be large enough to produce labels that can be used to train a CEM to recognise the discovered concept.

Once we have chosen which embedding set to use, we create labels for the new concept. Let c be the chosen concept and b be **true** if we decided to cluster embeddings from examples where c is predicted to be present, and **false** otherwise. We label the examples in E_c^b as follows: all the examples in the largest cluster are marked as having the new concept. The remaining examples

are ordered by distance from the centre of the largest cluster, a buffer of examples is marked as "unsure", and the farthest examples from the largest cluster are marked as not exhibiting the new concept. The size of the "unsure" buffer is controlled by hyperparameters. We set its size to be 70% of the size of the embedding set.

The examples corresponding to embeddings in $E_c^{\neg b}$ can either be marked as not exhibiting the concept, or be marked as "unsure". The former option is good when we are discovering sub-concepts of c, whose presence depends on c's. The latter option is more appropriate if we are not discovering sub-concepts. This choice is a hyperparameter.

The generated concept labels are compared to the labels for all the existing concepts. If the new labels agree with another concept's labels in a high proportion of examples, the discovered concept is deemed to be a duplicate, and we try clustering E_c^b again using a different random seed. This is because, with a new random seed, the largest cluster could be different, allowing us to discover a new concept. We try this up to ρ times (a hyperparameter), and if a new concept cannot be discovered, we move on to the next most desirable set of embeddings (as decided by silhouette scores).

Once an acceptable set of concept labels has been discovered, a new CEM is trained, with the discovered concept labels added to the training dataset. We train the new CEM from scratch, not reusing the components of the previous CEM. This is because the previous CEM's concept embeddings encoded information about the discovered concept. However, now that the discovered concept is included in the new iteration of the CEM, we hypothesise that the other concept embeddings will not need to learn to encode it, and may instead encode other concepts that we can discover.

In a second de-duplication check, the predictions for the discovered concept on the training dataset are compared with the predictions for the other concepts. If the new concept predictions agree with another concept's in a high proportion of examples, we discard the new model and move on to cluster the next most desirable set of embeddings, in the hope that this will allow us to discover a concept that is not already included in the concept annotations.

The process of discovering concepts can be repeated until either a specified maximum number of concepts has been discovered, or no more concepts can be discovered because all possible sets of embeddings have been clustered and none lead to a new concept. Pseudocode for our Concept Excavation algorithm can be found in Appendix A, and the hyperparameters are summarised in Appendix B.

An interpretation can be imposed on a discovered concept by prototyping: providing training examples that strongly activate the concept. This is the same method used by Alvarez-Melis and Jaakkola [14] to give meanings to discovered concepts. When evaluating Concept Excavation, we automatically match each discovered concept to a human-understandable concept in a "concept bank" (that was not used when training the initial model).

4. Experiments

In this section, we evaluate Concept Excavation by exploring the following research questions:

1. (RQ1) Does Concept Excavation discover interpretable concepts?

2. (RQ2) Does introducing the concepts discovered with Concept Excavation have an effect on the task and concept accuracies of the original CEM?
3. (RQ3) Can task accuracy be improved by intervening on concepts discovered with Concept Excavation?

4.1. Setup

Metrics For each task, we discover up to 10 new concepts. We limit the number of concepts discovered so that our experiments complete in a reasonable amount of time, however in real-world use cases more than 10 concepts could be discovered. We report the accuracies of the models as more concepts are introduced. Whenever we measure concept accuracy, we use the area under the receiver operating characteristic curve to avoid being misled by a majority-class classifier. We evaluate the effect of concept interventions by measuring the models' task accuracies as more concepts are intervened. All metrics in our evaluation are computed on test datasets using three random seeds, from which we compute each metric's mean and standard deviation.

As in [15], we measure the discovered concept accuracies, and perform concept interventions, by automatically matching each discovered concept to a human-understandable "left-out" concept. For each dataset, we have a "concept bank" containing concepts that are not used to train the initial CEM but that we anticipate might be discovered (e.g., "the first digit is 1" in our MNIST-ADD task, and "the shape is in the top left quadrant" in our dSprites task). Discovered concepts are matched with concepts in this concept bank using the concept predictions for the discovered concept on the training dataset. If a discovered concept does not match one in the concept bank, then it is discarded.

Datasets We evaluate our method using four datasets: MNIST-ADD [16] (handwritten digits), dSprites [17] (procedurally generated shapes), Caltech-UCSD Birds-200-2011 (CUB) [18] and Animals with Attributes 2 (AwA) [19]. Further details can be found in Appendix C.

Baselines We compare the models produced with Concept Excavation to vanilla CEMs and CBMs. We compare the accuracy of our models to black box DNNs to evaluate the effect of concept supervision. We use the same architecture capacity across all models. Appendix D contains further details on the model architectures and training hyperparameters.

Model Selection The hyperparameter values used for Concept Excavation and competing baselines, along with a discussion on how they were chosen, can be found in Appendix B.

4.2. Task and concept accuracy

In all our tasks, Concept Excavation was able to consistently discover 10 new concepts. The vast majority of discovered concepts were successfully matched to an interpretation using our concept banks. Details are contained in Appendix E. We note that Concept Excavation involves training many iterations of the model, and it is therefore computationally expensive: for example, it takes around 14 hours to discover 10 concepts in the AwA task.

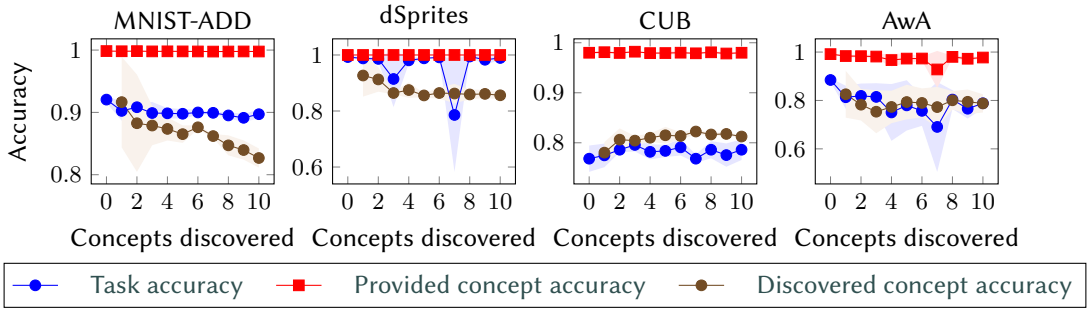

Figure 2: Accuracy of CEMs as concepts are discovered and added to their training dataset. Discovered concepts are predicted with high accuracy, and task and provided concept accuracies remain high when discovered concepts are introduced.

Discovered concepts have human-interpretable interpretations and can be predicted accurately (RQ1). The vast majority of concepts discovered were matched to a human-interpretable concept in the concept bank. For example, the meaning assigned to one of the concepts discovered on the third run on the MNIST-ADD dataset was "the second digit is 1". Figure 2 shows that the discovered concept accuracies, evaluated using the ground truth labels of the corresponding human-interpretable concepts on the test datasets, are high, exceeding 90% in some cases. This demonstrates that the concept labels produced by Concept Excavation are sufficient to teach the CEM to recognise the discovered concept accurately.

Introducing concepts discovered with Concept Excavation does not have a large effect on task accuracy or provided concept accuracy in most cases (RQ2, Figure 2). The exception to this is the AwA task, where the task accuracy decreases by 10% after 10 concepts have been introduced. The dips in task accuracy on the dSprites task when three and seven concepts have been introduced come from a single anomalous run. We believe they are noise and not caused by the introduction of discovered concepts, because they were only present in one run, and the task accuracy recovers after the next concept has been introduced. Appendix F compares the task and concept accuracy of CEMs after Concept Excavation with our baselines. In all the datasets but AwA, our models have task accuracies that are competitive with CEMs and sometimes even black box DNNs. The mean concept accuracy of our models is generally slightly lower than that of CEMs and CBMs, however our models are predicting discovered concepts that were not provided as part of the initial training dataset. Overall, the effect of Concept Excavation on the task and provided concept accuracy is insignificant in most cases, so the additional interpretability offered by the discovered concepts does not come at the cost of accuracy.

4.3. Concept interventions

While intervening on provided concepts improves task accuracy, interventions on discovered concepts do not (RQ3). As shown in Figure 3, intervening on provided concepts leads to an increase in task accuracy. However, interventions on the discovered concepts lead to

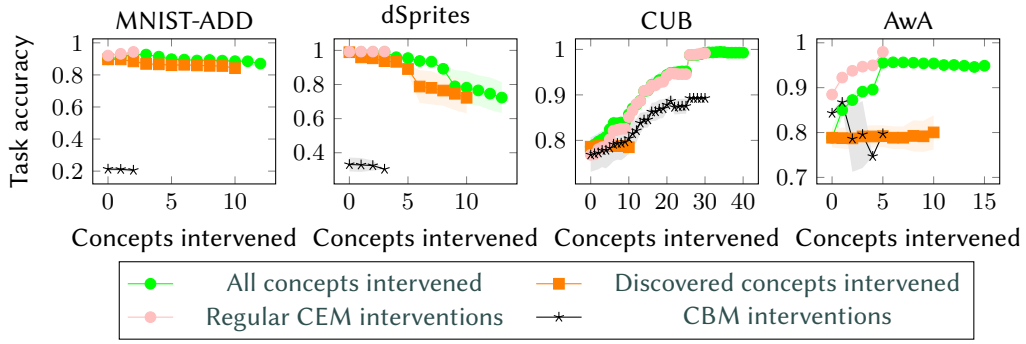

Figure 3: Accuracy as concepts are intervened. When all concepts are intervened on, we first intervene on the provided concepts and then on the discovered concepts, in the order they were discovered. When only the discovered concepts are intervened on, they are intervened on in the order they were discovered.

at most minor improvements in task accuracy, and can sometimes decrease task accuracy. We hypothesise that this is because the discovered concept labels are not as accurate as the ground truth labels in the dataset. We elaborate on this in Appendix G and suggest it for future work.

5. Discussion

Limitations and future work The main limitation of Concept Excavation that could be addressed by future work is that the discovered concepts do not respond well to concept interventions. This might be because the discovered concepts capture information beyond the human-understandable interpretation we assign them. For example, in our MNIST-ADD task, one discovered concept was assigned the meaning "the second digit is one", but the first digit was zero in a surprisingly small proportion of examples labelled as having the concept. Despite this, our experiments show that the discovered concepts can be predicted with very high accuracy.

Additionally, we note that Concept Excavation is computationally expensive: it took an average of 14 hours to discover 10 concepts in the AwA task. However, it only needs to be run once per task, and the maximum number of concepts to discover can be set so that the running time is acceptable.

Conclusion Our experiments show that Concept Excavation discovers interpretable concepts that can be introduced into CEMs without sacrificing task or provided concept accuracy in most cases. After they have been introduced, the discovered concepts can be predicted with high accuracy. This addresses a major drawback of CEMs: the need for large numbers of concept labels in their training dataset. While there is room for improvement in allowing effective concept interventions on discovered concepts and resource utilisation while discovering concepts, Concept Excavation reduces the barriers to using CEMs in real-world scenarios.

Acknowledgments

MEZ acknowledges further support from the Gates Cambridge Trust via a Gates Cambridge Scholarship.

References

[1] A. Barredo Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. Garcia, S. Gil-Lopez, D. Molina, R. Benjamins, R. Chatila, F. Herrera, Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai, Information Fusion 58 (2020) 82–115. URL: https://www.sciencedirect.com/science/article/pii/S1566253519308103. doi:https://doi.org/10.1016/j.inffus.2019.12.012.

[2] M. Espinosa Zarlenga, P. Barbiero, G. Ciravegna, G. Marra, F. Giannini, M. Diligenti, Z. Shams, F. Precioso, S. Melacci, A. Weller, P. Lió, M. Jamnik, Concept embedding models: Beyond the accuracy-explainability trade-off, in: Advances in Neural Information Processing Systems, volume 35, Curran Associates, Inc., 2022, pp. 21400–21413. URL: https://proceedings.neurips.cc/paper_files/paper/2022/file/867c06823281e506e8059f5c13a57f75-Paper-Conference.pdf.

[3] D. Bau, B. Zhou, A. Khosla, A. Oliva, A. Torralba, Network dissection: Quantifying interpretability of deep visual representations, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE Computer Society, Los Alamitos, CA, USA, 2017, pp. 3319–3327. URL: https://doi.ieeecomputersociety.org/10.1109/CVPR.2017.354. doi:10.1109/CVPR.2017.354.

[4] R. Fong, A. 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), IEEE Computer Society, Los Alamitos, CA, USA, 2018, pp. 8730–8738. URL: https://doi.ieeecomputersociety.org/10.1109/CVPR.2018.00910. doi:10.1109/CVPR.2018.00910.

[5] B. Kim, M. Wattenberg, J. Gilmer, C. Cai, J. Wexler, F. Viegas, R. 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*, PMLR, 2018, pp. 2668–2677. URL: https://proceedings.mlr.press/v80/kim18d.html.

[6] A. Ghorbani, J. Wexler, J. Y. Zou, B. Kim, Towards automatic concept-based explanations, in: Advances in Neural Information Processing Systems, volume 32, Curran Associates, Inc., 2019. URL: https://proceedings.neurips.cc/paper_files/paper/2019/file/77d2afcb31f6493e350fca61764efb9a-Paper.pdf.

[7] C.-K. Yeh, B. Kim, S. Arik, C.-L. Li, T. Pfister, P. Ravikumar, On completeness-aware concept-based explanations in deep neural networks, in: H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, H. Lin (Eds.), Advances in Neural Information Processing Systems, volume 33, Curran Associates, Inc., 2020, pp. 20554–20565. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/ecb287ff763c169694f682af52c1f309-Paper.pdf.

[8] M. Espinosa Zarlenga, K. Collins, K. Dvijotham, A. Weller, Z. Shams, M. Jamnik, Learning to receive help: Intervention-aware concept embedding models, Advances in Neural Information Processing Systems 36 (2024).

[9] T. P. Oikarinen, S. Das, L. M. Nguyen, T. 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.

[10] P. W. Koh, T. Nguyen, Y. S. Tang, S. Mussmann, E. Pierson, B. Kim, P. Liang, Concept bottleneck models, in: Proceedings of the 37th International Conference on Machine Learning, volume 119 of *Proceedings of Machine Learning Research*, PMLR, 2020, pp. 5338–5348. URL: https://proceedings.mlr.press/v119/koh20a.html.

[11] L. van der Maaten, G. Hinton, Visualizing data using t-SNE, Journal of Machine Learning Research 9 (2008) 2579–2605. URL: http://jmlr.org/papers/v9/vandermaaten08a.html.

[12] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, O. Wang, The unreasonable effectiveness of deep features as a perceptual metric, in: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), IEEE Computer Society, Los Alamitos, CA, USA, 2018, pp. 586–595. URL: https://doi.ieeecomputersociety.org/10.1109/CVPR.2018.00068. doi:10.1109/CVPR.2018.00068.

[13] P. J. Rousseeuw, Silhouettes: A graphical aid to the interpretation and validation of cluster analysis, Journal of Computational and Applied Mathematics 20 (1987) 53–65. URL: https://www.sciencedirect.com/science/article/pii/0377042787901257. doi:https://doi.org/10.1016/0377-0427(87)90125-7.

[14] D. Alvarez Melis, T. Jaakkola, Towards robust interpretability with self-explaining neural networks, in: S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, R. Garnett (Eds.), Advances in Neural Information Processing Systems, volume 31, Curran Associates, Inc., 2018. URL: https://proceedings.neurips.cc/paper_files/paper/2018/file/3e9f0fc9b2f89e043bc6233994dfcf76-Paper.pdf.

[15] M. E. Zarlenga, Z. Shams, M. E. Nelson, B. Kim, M. Jamnik, Tabcbm: Concept-based interpretable neural networks for tabular data, Transactions on Machine Learning Research (2023).

[16] Y. LeCun, C. Cortes, C. Burges, Mnist handwritten digit database, Online, 2010. URL: http://yann.lecun.com/exdb/mnist.

[17] L. Matthey, I. Higgins, D. Hassabis, A. Lerchner, dsprites: Disentanglement testing sprites dataset, 2017. URL: https://github.com/deepmind/dsprites-dataset/.

[18] C. Wah, S. Branson, P. Welinder, P. Perona, S. Belongie, The Caltech-UCSD Birds-200-2011 Dataset, Technical Report CNS-TR-2011-001, 2011.

[19] Y. Xian, C. H. Lampert, B. Schiele, Z. Akata, Zero-shot learning—a comprehensive evaluation of the good, the bad and the ugly, IEEE Transactions on Pattern Analysis and Machine Intelligence 41 (2019) 2251–2265. doi:10.1109/TPAMI.2018.2857768.

[20] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778. doi:10.1109/CVPR.2016.90.

[21] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, L. Fei-Fei, Imagenet large scale visual recognition

Concept excavation

Input: CEM m and training dataset d
Output: CEM m' and dataset d' with a discovered concept
Hyperparameter: ν, minimum embedding set size
Hyperparameter: γ, number of clusters
Hyperparameter: β, deduplication check similarity threshold

1: $S \leftarrow \texttt{calculateEmbeddings}(m)$
2: $S \leftarrow \texttt{removeSmallSets}(S, \nu)$
3: $E \leftarrow \arg\max_{e \in S}(\texttt{SilhouetteScore}(\texttt{KMeans}(e, \gamma)))$
4: $\texttt{labels} \leftarrow \text{LABEL}(E, d)$
5: **if** $\texttt{labels} == \emptyset$:
6: remove E from S
7: **if** S is not empty **then goto** line 3 **else fail**
8: $d' \leftarrow d$ with new concept annotations from $\texttt{labels}$
9: $m' \leftarrow \texttt{trainCem}(d')$
10: $t = \texttt{max}$(proportion of examples where new concept's prediction agrees
11: with g's prediction for all other concepts g in d')
12: **if** $t < \beta$:
13: **return** m', d'
14: **else** :
15: remove E from S
16: **if** S is not empty **then goto** line 3 **else fail**

Note: In line 1, `calculateEmbeddings` returns a set of sets of embeddings. For each concept c in m, there are two disjoint sets of embeddings: the first contains c's embeddings in all examples where c is predicted to be present. The second contains c's embeddings in all examples where c is not predicted to be present.

Figure 4: Pseudocode for concept excavation. This uses the LABEL subroutine, defined in Figure 5.

challenge, International Journal of Computer Vision 115 (2015) 211–252. URL: https://doi.org/10.1007/s11263-015-0816-y. doi:10.1007/s11263-015-0816-y.

[22] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, in: Y. Bengio, Y. 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.

A. Concept Excavation pseudocode

The pseudocode for the concept excavation algorithm is given in Figure 4. This makes use of the LABEL subroutine, the pseudocode for which is given in Figure 5.

B. Hyperparameters

The hyperparameters of Concept Excavation are listed in Table 1, along with the values we used in our experiments.

LABEL subroutine

Input: Set of embeddings E and training dataset d
Output: Discovered concept labels `labels`, or $\emptyset$
Hyperparameter: ρ, number of times to retry clustering
Hyperparameter: γ, number of clusters
Hyperparameter: κ, proportion of embeddings set that should be labelled as "unsure"
Hyperparameter: ψ, minimum proportion of embeddings set that should be labelled as "off"
Hyperparameter: χ, whether we expect to discover sub-concepts
Hyperparameter: α, deduplication check similarity threshold

```
1:  for i in [1, 2, 3, ..., ρ] :
2:      clusters ← KMeans(E, γ)
3:      c ← largest cluster in clusters
4:      bufferLen ← ⌊|E| × κ⌋
5:      numberOff ← |E| − (|c| + bufferLen)
6:      if numberOff < |E| × ψ :
7:          numberOff ← min(⌊|E| × ψ⌋ , |E| − |c|)
8:      labels ← repeat(UNSURE, |d|)
9:      labels: mark examples whose embeddings are in c as "on"
10:     labels: mark the examples farthest from the centre of c as
11:             "off". The number of examples to mark as "off" is
12:             given by numberOff.
13:     if χ :
14:         labels: mark examples whose embeddings are not in E
15:             as "off"
16:     t = max(proportion of labels that agree with g's labels for all
17:             concepts g in d)
18:     if t < α :
19:         return labels
20: return ∅
```

Note: A different random initialisation is used each time line 2 is executed.

Figure 5: Pseudocode for the LABEL subroutine.

The value of γ, as well as the decision to use the K-Means clustering algorithm, was made by trying different values and algorithms and visualising the results on the training datasets (using t-SNE [11]). We also measured the mutual information between the clusters and the ground truth concept labels from the concept bank (again on the training datasets). The test sets were not used as part of this process. The clustering step in the ACE algorithm [6] also uses the K-Means algorithm with 25 clusters.

The smallest value we could have chosen for ν is 25, as to run the clustering algorithm we need at least as many examples as clusters. To produce meaningful discovered concept labels, we need more than one example per cluster, so we set ν higher than 25. We did not attempt to fine-tune the value, and did not run into problems with our chosen value ($\nu = 100$). If Concept Excavation returns too many "unsure" labels, the value of ν could be increased.

The values of κ and ψ were chosen using visualisations on the training datasets, without any use of the test datasets.

Table 1
Concept Excavation hyperparameters.

Hyperparameter	Value	Description
χ	Task dependent	Whether we expect to discover sub-concepts
γ	25	Number of clusters
ν	100	Minimum embedding set size
κ	0.7	Proportion of embeddings set that should be labelled as "unsure"
ψ	0.2	Minimum proportion of embeddings set that should be labelled as "off"
ρ	10	Number of times to retry clustering
α	0.9	First deduplication check similarity threshold
β	0.8	Second deduplication check similarity threshold

We did not fine-tune the value of ρ; we realised when experimenting on the training dataset that running the clustering with a different random seed sometimes led to a new concept being discovered when previously a duplicate was discovered, which is why the clustering is retried if a new concept is not discovered the first time.

The values of α and β were chosen by running Concept Excavation on the training dataset and selecting values that led to few duplicate concepts being discovered. Duplicates were identified by inspecting the discovered concept labels returned by Concept Excavation (for the training dataset) to determine human-interpretable meanings for the discovered concepts.

The value of the χ hyperparameter is chosen based on whether we expect to discover sub-concepts. In the MNIST-ADD, dSprites and CUB tasks we set $\chi =$ **true** because we anticipate that the discovered concepts will be sub-concepts of the provided ones. In the AwA task we set $\chi =$ **false** because we do not expect the discovered concepts to be sub-concepts of the provided ones. In practice, the choice of χ could be guided by inspecting the human-interpretable meanings assigned to discovered concepts.

We found that clustering the embeddings for a single concept rather than concatenating the embeddings for multiple concepts gave better results in practice. It allows sub-concepts to be discovered: for instance, in the MNIST-ADD dataset, in one case clustering the concept embeddings for the "second digit is greater than three" concept, taken from examples where it was active, led to the discovery of the "second digit is five" concept, and there are multiple similar examples. However, the concepts discovered when clustering the embeddings for an existing concept are not always sub-concepts of the existing one.

C. Datasets and tasks

C.1. MNIST-ADD

Examples in the MNIST-ADD dataset are formed by concatenating images of handwritten digits from the MNIST dataset [16].

Examples in MNIST-ADD contain two handwritten digits, each between 0 and 6 (inclusive). The label is the sum of the digits. There are two concepts: the first one indicates whether the

first digit is greater than three, and the second one indicates whether the second digit is greater than three. The examples have shape $2 \times 28 \times 28$, with each digit residing in its own channel. Each grayscale pixel is represented by a scalar value between 0 and 1. The concepts in the concept bank, which are used to assign a human-interpretable meaning to discovered concepts, consist of a one-hot representation of the digits (for example one of the concepts is "the first digit is 3").

MNIST-ADD is split into train, validation and test sets. The train and test sets each contain 10,000 examples, and the validation set contains 2,000 examples. To generate them, the MNIST training set is randomly divided into a training set and a validation set (with 80% of the examples in the training set and 20% in the validation set). We, therefore, have three sets of digits: the training and validation sets (created from the MNIST training set) and the MNIST test set. To generate one of the MNIST-ADD sets, the appropriate MNIST digit set is filtered so it contains only the desired digits, and then examples are created by concatenating randomly chosen digits from the filtered set (with replacement).

C.2. dSprites

The dSprites dataset [17] consists of 2D shapes procedurally generated from all possible combinations of 6 ground truth independent latent factors. The factors are colour, shape, scale, orientation, x position, and y position. The black-and-white images are all 64×64 pixels in size.

In our experiments, only shapes that have not been rotated (that is, shapes where the orientation latent factor is 0) are used, and the dataset is randomly divided into train, validation and test splits of sizes 11,059, 2,765 and 4,608 respectively. Pixel values are scaled to lie in the range [0, 1].

For each image, the quadrant the shape is in is calculated from the x and y position latent factors. The quadrant is represented by an integer in the range [0, 3]. The shape in the image (square, ellipse or heart) is represented by an integer in the range [0, 2]. The scale is represented by an integer in the range [0, 5]. The label of an example is given by the sum of the integers representing its quadrant, shape and scale. There are three concepts: the first indicates whether the shape is in the right half of the image, the second indicates whether the shape is a square and the third indicates whether the shape is "big" (its scale value is one of the largest three) or "small" (its scale value is one of the smallest three). The concepts in the concept bank form one-hot representations of the quadrant, shape, and scale latent factors (for example, one of the concepts is "the shape is in the top left quadrant").

C.3. Caltech-UCSD Birds-200-2011

CUB is a dataset of images of 200 bird species [18].

We use the same preprocessing as Koh et al. [10]. Because the binary attributes are noisy, they are denoised with majority voting; for example, if more than half of mockingbirds have black wings in the data, then all mockingbirds are labelled as having black wings. Binary attributes that are present in fewer than 10 classes are removed. We use the train, validation and test splits provided by Koh et al. [10].

For our experiments, we randomly choose 20 of the 200 bird species, and remove all examples that do not belong to one of the 20 chosen species. Each image is labelled with the species of the bird it contains.

Some of the binary attributes correspond to the colour of a particular part of the bird, for example "has yellow wings" or "has black wings". We classify the colours used as either "light" or "dark". For each body part for which there are binary attributes indicating its colour, we introduce two concepts: for example, for wings, we introduce the concepts "has light wings" and "has dark wings". This leads to 30 concept annotations for each example, corresponding to 15 body parts. The concept bank consists of all the binary attributes (after preprocessing).

We adopt the preprocessing used by Koh et al. [10]. During training, the brightness and saturation of the images are randomly altered, and then a random portion of the image is cropped and it is resized to 299×299 pixels. Some randomly chosen images are horizontally flipped. Each channel of the image is normalised to have mean 0.5 and standard deviation 2. During testing, the images are cropped in the centre to have size 299×299 pixels. Each channel of the image is normalised to have mean 0.5 and standard deviation 2.

C.4. Animals with Attributes 2

The AwA dataset consists of images of 50 animal species [19]. Each image is labelled with 85 binary attributes. The attribute values are determined by the image's class, so all images in the same class have the same attribute labels.

In our AwA task we randomly select 10 out of the 50 animal species and only use images belonging to one of the selected species. We discard attributes that are not visual (for example "smelly") and use five of the remaining attributes as concept labels. The concept bank consists of all the visual binary attributes.

The same transformations are applied during training and testing as are applied to the CUB images.

D. Model architectures and training

For each task, we train an initial CEM and then apply Concept Excavation to train new iterations with more concepts. All the CEMs trained for the same task have the same architecture, although they have different numbers of concepts. For each task, we also train a CBM and a black box model to use as baselines.

For the MNIST-ADD and dSprites tasks, the DNN ψ that produces the latent representation that is input to the embedding generators of the CEM [2] comprises four convolutional layers whose parameters are given in Table 2. The input to each convolutional layer is padded so that the output of each layer is the same size as its input. Each convolutional layer is followed by a batch normalisation layer and a leaky rectified linear unit (LeakyReLU) activation function. The convolutional layers are followed by a fully connected layer with 128 outputs. Each embedding generator consists of a single fully connected layer with 128 inputs and 16 outputs. The scoring function, which assigns a probability to each concept, is shared between all concepts and starts with a single fully connected layer with 32 inputs (corresponding to the concatenated positive and negative embeddings for a concept) and a single output. The fully connected layer is

Table 2
Convolutional layers in CEMs for MNIST-ADD and dSprites

Layer	Input channels	Output channels	Kernel size
1	2 (MNIST-ADD), 1 (dSprites)	16	3×3
2	16	16	3×3
3	16	16	3×3
4	16	16	3×3

followed by a sigmoid function. The label predictor consists of three fully connected layers. The first and second layers have 128 outputs and are followed by LeakyReLU activation functions, and the number of outputs in the final layer is determined by the number of tasks.

In the baseline CBM, the concept encoder [10] has the same architecture as the module ψ in the CEM, except that the number of outputs in the final fully connected layer is equal to the number of concepts. A sigmoid function is applied after the final layer to get the concept probabilities, which are input to the label predictor. The label predictor has the same architecture as the label predictor in the CEM, except the number of inputs to the first layer is equal to the number of concepts. The black box baseline has the same architecture as the CBM, except 128 activations are used in the "concept bottleneck", no sigmoid function is used and there is no concept supervision during training.

For the CUB and AwA tasks, the DNN ψ in the CEM is a ResNet-34 [20] model that has been pretrained on the ImageNet dataset [21]. The embedding generators, scoring functions and label predictors are the same as for the MNIST-ADD and dSprites tasks. In the baseline CBM, the concept encoder is a ResNet-34 model, again pretrained on the ImageNet dataset, followed by a fully connected layer whose number of outputs is equal to the number of concepts. The fully connected layer is followed by a sigmoid function. The label predictor is the same as in the MNIST-ADD and dSprites tasks. The black box baseline has the same architecture as the CBM, except 1000 activations are used in the "concept bottleneck" and there is no fully connected layer appended to the ResNet-34 model in the "concept encoder".

The models are trained using the Adam optimisation algorithm [22] with a learning rate of 1×10^{-3}. The models for the MNIST-ADD and dSprites tasks are trained for a maximum of 300 epochs, and the models for the CUB and AwA tasks are trained for a maximum of 150 epochs. For all models, training is stopped if the validation loss does not improve for 25 epochs. We use a batch size of 2048 when training the MNIST-ADD models, 256 when training the dSprites models, and 128 when training the CUB and AwA models. When training the CEMs, the RandInt regularisation strategy is used [2]. At train-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. [2] find that it enables effective interventions while giving good performance.

Table 3
Concept Excavation statistics

	MNIST-ADD	dSprites	CUB	AwA
Discovered	10 ± 0	10 ± 0	10 ± 0	10 ± 0
No match in bank	0 ± 0	0 ± 0	0 ± 0	1 ± 0
Duplicates rejected	18 ± 2	41 ± 1	217 ± 33	9 ± 3
Duplicates missed	2 ± 1	3 ± 1	2 ± 0	2 ± 1
Execution time (hours)	0.82 ± 0.20	3.0 ± 1.2	10 ± 5.4	14 ± 11

E. Concept Excavation statistics

Statistics on the executions of Concept Excavation are summarised in Table 3: for each task, we report the number of concepts discovered, the number of concepts that were discarded because they did not find a match in the concept bank, the number of concepts that were discarded by a deduplication check, the number of duplicate concepts in the final set of discovered concepts and the execution time.

F. Comparison to baselines

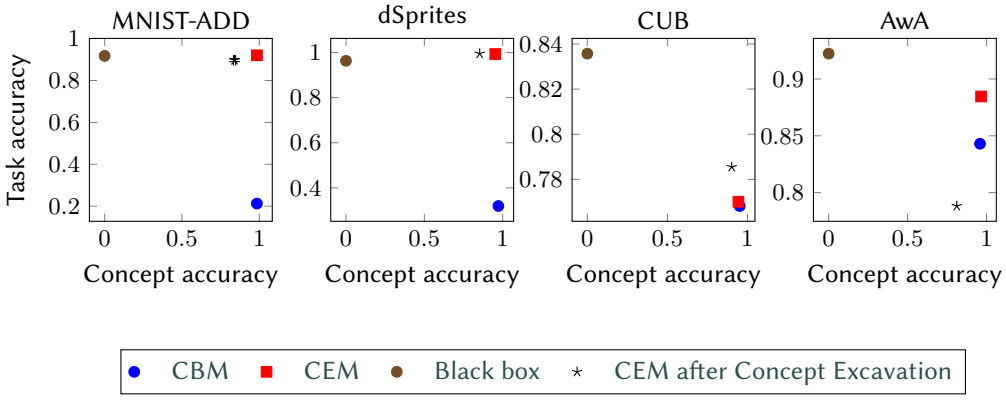

Figure 6: Comparison of the accuracies of CEMs after Concept Excavation to baselines. The task accuracy of CEMs after Concept Excavation is mostly competitive with that of vanilla CEMs. The mean concept accuracy of CEMs after Concept Excavation is generally slightly lower than that of CEMs and CBMs, but after Concept Excavation the CEMs are predicting discovered concepts that were not part of the initial training dataset.

Figure 6 compares the task and overall mean concept accuracies of CEMs after concept excavation with our baselines. In all the datasets but AwA, our models have task accuracies that are competitive with CEMs and sometimes even black box DNNs. The mean concept accuracy of our models is generally slightly lower than that of CEMs and CBMs, however our models are predicting discovered concepts that were not provided as part of the initial training dataset.

The poor task accuracy of CBMs in MNIST-ADD and dSprites is likely because the concept annotations provided with the training dataset do not contain enough information to accurately predict the label.

G. Exploring discovered concept labels

Figure 7 displays the distribution of digit values in examples labelled as having a discovered concept, and in the entire training dataset, for the MNIST-ADD task. The human-interpretable meaning assigned to the discovered concept is "the second digit is 1". While this is broadly consistent with the distribution of digit values in examples labelled as having the concept, there are some peculiarities. For instance, the first digit is 0 in a surprisingly small proportion of the examples labelled as having the concept, and the second digit is 3 in about 10% of examples labelled as having the concept. Addressing this would be an interesting problem for future work to explore.

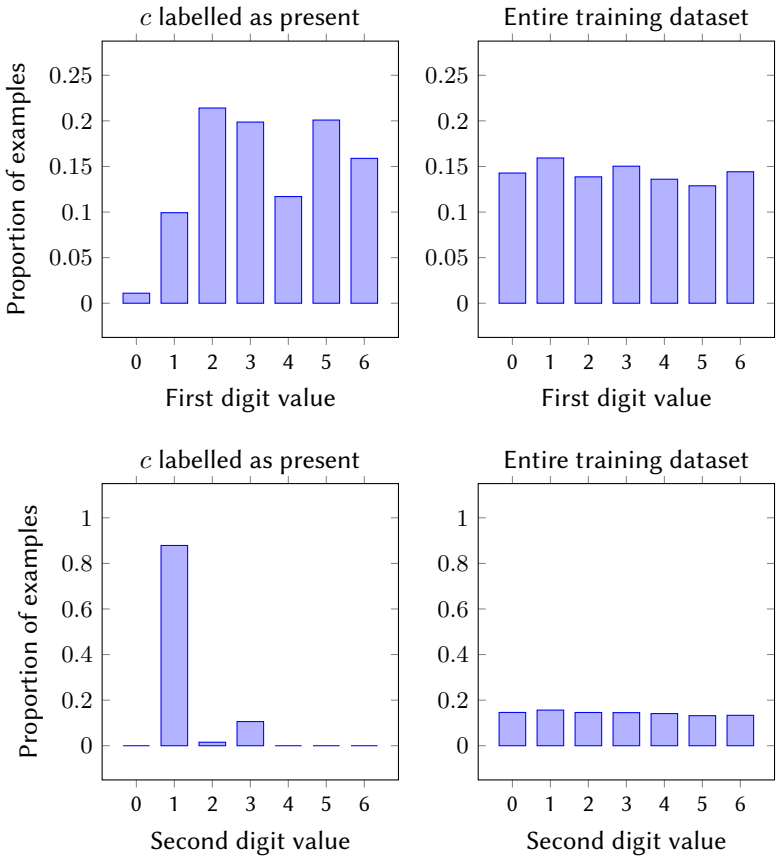

Figure 7: Distribution of digit values in examples labelled as having discovered concept c. c was the first concept discovered on the third run on the MNIST-ADD dataset. The human-interpretable meaning assigned to c is "the second digit is 1". The distribution of digit values in the entire training dataset is shown for comparison. While the distribution of digit values in examples labelled as having c are broadly consistent with its human-interpretable meaning, there are some anomalies. For example, the second digit is 3 in about 10% of the examples labelled as having c.