Measuring the (Un)Faithfulness of Concept-Based Explanations

Supplementary Material

This supplementary section provides information not included in the main text due to space constraints. It includes:

1. Sec. A: Discussion of related SAE faithfulness metrics.
2. Sec. B: Implementation details for the measure-over-measure study (Sec. 6.1).
3. Sec. C: Experimental details for the evaluation of U-CBEMs done in Sec. 6.2.
4. Sec. D: Results on additional models for the experiment done in Sec. 6.2.

A. Discussion and Comparison with SAE Faithfulness Metrics

There has been a significant effort in recent years to explore using SAEs for explainability, particularly on large language models [5]. SAEs are typically trained to minimize embedding reconstruction error (e.g., mean squared error—MSE), which is the primary metric used to evaluate their faithfulness. The intuition is that if an SAE can accurately reconstruct the intermediate embedding it is explaining, then downstream operations on the reconstructed embedding should also be faithfully preserved. While reconstruction error is useful, it is a *proxy* for faithfulness; there is no clear mapping from reconstruction to downstream prediction error, making it hard to interpret the significance of a given MSE error. To accommodate this, recent work [16, 29] introduces downstream losses on model outputs to measure faithfulness. Common choices are KL divergence and cross entropy loss. However, we argue that these are uninterpretable (as acknowledged in [16]). These metrics work well when used *relatively*, which fits the referenced papers's need to compare different SAEs, but these metrics do not give an *absolute* measure of faithfulness (unlike SURF), which is necessary when one wants to make an independent assessment of a method's faithfulness.

Similarly, [31] develops a new SAE architecture (Patch-SAE) and evaluates downstream classification accuracy after masking SAE latents (which is equivalent to a deletion-based approach). This metric suits their purpose, since they use SAEs to analyze model behavior, rather than to explain a prediction. [13] introduce a more robust SAE architecture (Archetypal SAEs) and introduce several metrics to evaluate. The closest metric to SURF is a plausibility metric, which measures the average maximum cosine similarity between the discovered CAVs and each classification vector. However, we argue that this metric does not give a holistic picture of faithfulness, since it only looks at the *maximum* and it is unclear how this metric maps on to errors in the output space (e.g., how does it get transformed by a softmax). We believe that SURF is a complementary measure that can be used to help study faithfulness of SAEs applied to vision models with the intention of explaining the model's output computation.

B. Implementation Details for Measure-over-Measure Comparison

This section contains details for the measure-over-measure comparison conducted in Sec. 6.1.

B.1. Experimental Setting Details

In the *Perfect* setting, we set CAVs and importances according to the weights of the final classification layer. Specifically, each class has 1 CAV ($K = 1$). The CAV $\mathbf{v_{c,1}}$ for class c is set to $\frac{\mathbf{f}_c}{\|\mathbf{f}_c\|_2}$, where $\mathbf{f}_c$ is class c's classification vector. The CAV's importance $\alpha_{c,1}$ is set to $\|\mathbf{f}_c\|_2$.

For the *random* settings, we sample each dimension of the CAVs independently from a standard normal distribution, and then make the CAV unit norm. We sample concept importances uniformly from $[0, 1)$.

B.2. Surrogate Implementation

Recall that U-CBEMs define their own mechanism $\mathcal{P} : \mathcal{H} \to \mathbb{R}^K$ to project embeddings to the concept basis. For these experiments, since we are not using any U-CBEM and instead directly setting the CAVs and importances, we define $\mathcal{P}(\mathbf{h}_j; V_i)_k = \mathbf{h}_j^T \mathbf{v}_{i,k}$, where i denotes the class, k denotes the CAV, and j selects the spatial embedding. Recall in this case that $K = 1$ and there is only one embedding (obtained from global pooling).

B.2.1. SURF Surrogate (Ours)

Following Sec. 5.1 and Eq. (10), we set the parameters of the surrogate with the obtained concepts and importances. This surrogate has no learnable parameters and is exactly equal to the original linear layer (up to the bias term). Note that the embeddings are linearly projected onto the CAVs to find the concept representation. After the surrogate reconstructs the output, we add the constant bias term before evaluation.

B.2.2. ICE Surrogate

ICE's surrogate takes the concept representation and reconstructs the embedding, as specified by the U-CBEM. It then passes through the final classification layer to obtain outputs. Let $\mathbf{p_i} \in \mathbb{R}^K$ denote the concept representation for class i obtained from embedding $\mathbf{h}$. The surrogate is:

$$y_i = \mathbf{f}_i^T \psi_{ICE}(\mathbf{p_i}) + b_i \tag{13}$$

where $\psi_{ICE}(.)$ denotes the reconstruction, as specified by the U-CBEM. Since we are not making use of any U-CBEM in the measure-over-measure comparison, we use linear operations to reconstruct the embedding from the concept representation. That is, $\psi(\mathbf{p_i}) = \sum_{k=1}^{K} p_{i,k}\mathbf{v}_{i,k}$.

B.2.3. C-SHAP Surrogate

C-SHAP's surrogate defines a two-layer MLP $\psi_{\text{C-SHAP}}(.)$ to reconstruct the embedding, and then passes through the final classification layer to obtain outputs. The surrogate is:

$$y_i = \mathbf{f}_i^T \psi_{\text{C-SHAP}}(\mathbf{p_i}) + b_i \tag{14}$$

Since C-SHAP's surrogate contains learnable parameters. Following [48], the MLP's hidden layer has a dimensionality of 500, and the surrogate is trained using Cross Entropy Loss between the surrogate's predicted class probabilities and the original model's class probabilities. We train the surrogate using the Adam optimizer [26] and a starting learning rate of 0.1. The surrogate is trained for at most 100 epochs, during which the learning rate is decayed by a factor of 0.5 when the training loss plateaus. We stop training when the learning rate is reduced below 10^{-7}.

B.3. FLOPs Computation

We treat multiply and add as 2 FLOPs. Let K be the number of concepts, C be the number of output classes, and D by the dimensionality of the embedding space.

SURF Surrogate: Given that the surrogate is a linear layer from concept representations to the output space (though it uses different concept representations for each class), the number of FLOPs required is $2KC$.

ICE Surrogate: This surrogate reconstructs the original embedding (by a linear projection) from the concept representation, and then uses the original model (which is also a linear projection) to map to the output space. Thus, the number of FLOPs required is $CD(2K + 1)$.

C-SHAP Surrogate: This surrogate reconstructs the original embedding (with a two-layer MLP) from the concept representation, and then uses the original model (which is a linear projection) to map to the output space. Thus, the number of FLOPs required is $2C(H_1K + H_1D + D)$, where H_1 is the dimensionality of the MLP's hidden layer.

C. Additional Details for Benchmarking U-CBEM Faithfulness

Sec. 6.2 evaluates prior U-CBEMs, finding that they produce unfaithful explanations. This section gives experimental and implementation details for this evaluation.

C.1. Model Details

Object Classification: Our evaluation uses a ResNet-50, available from the `timm` library. We use the ImageNet pre-trained weights and initialize the final classification layer with random weights. The model is finetuned (`conv4` and all subsequent layers) on the Caltech-101 training set, achieving a test accuracy of approximately 95.61%. The model is trained without data augmentation and using cross entropy loss with label smoothing.

Multi-Attribute Prediction: We finetune an ImageNet pre-trained MobileNetV2 on the CelebA dataset for the task of binary attribute prediction. We initialize the final prediction linear layer with random weights, and finetune all layers of the model. We achieve an attribute prediction accuracy of 91.55%. The model is trained with data augmentation and binary cross-entropy loss.

Age Regression: We finetune an ImageNet pre-trained Vision Transformer (ViT) on the UTK-Face dataset for the task of age regression. We choose the ViT-B/16 variant for our experiment. We initialize the final prediction linear layer with random weights, and finetune all weights above (not including) the 9th layer. The model achieves a mean average error of 5.14 on the test set. The model is trained with data augmentation and using L1 Loss.

C.2. Dataset Details

We learn each class's concepts and importances using images of the class from the training set. Importantly, the label of each image is determined using the original model's prediction (instead of the ground truth label). We do this because we are interested in explaining why *the model* predicts a specific label, regardless of whether the prediction is correct or not. After learning the U-CBEMs, we evaluate them on a test set.

Object Classification: We split Caltech-101's dataset into train and test sets using 80/20 splits. While splitting the data, we maintain the same class distribution present in the original dataset in both the training and testing splits. To handle Caltech-101's class imbalance, we assemble a balanced test set; this ensures that we evaluate for faithfulness equally across all classes.

Multi-Attribute Prediction: We use CelebA's predefined train and test sets. We do not make any modifications to the test set. Since the training set is too large for many U-CBEMs, we take a random subset of training data to use for finding U-CBEM parameters. The random subset is equivalent in size to the test set.

Age Regression: We split the dataset into train and test sets using 80/20 splits. While splitting the data, we discretize the age distribution into bins of 10-years, and maintain this distribution across both the training and testing splits. As UTK-Face does not have any classes, all U-CBEMs find CAVs on all embeddings.

C.3. U-CBEM Details

This section specifies implementation details for each U-CBEM that was benchmarked.

C.4. CDISCO

The open-source implementation provided by the authors was used to evaluate this approach. CDISCO uses SVD to discover CAVs. Then, a gradient-based approach is used to calculate the importance of each CAV. The code used to compute the gradients was not provided in the repository, so we re-implemented it. Embeddings are linearly projected to the concept space.

Object Classification: Due to a lack of clarity, we had to make one implementation choice. Specifically, when finding the importance (using a gradient calculation), it was unclear if we should use images from all classes or only images of the CAV's class. We opted for the latter, based on the code and the example provided by the authors.

Multi-Attribute Prediction: CDISCO was extended to this task by reformulating the task as independent binary classification tasks. Thus, CAVs and concept importances were learned for each task independently

Age Regression: The open-source implementation only considered the case of multi-class classification. According to the paper and the code used for multi-class classification, we re-implemented CDISCO to work in the case of single-output regression.

C.5. ICE

The open-source implementation provided by the authors was used to evaluate this approach. ICE uses NMF to find CAVs. Due to the potentially large number of embeddings, we use the batched-version of NMF. Concept importance is found using the TCAV method proposed in [24]. Embeddings are projected to the concept space using sci-kit learn's transform method. ICE was extended to multi-attribute prediction by reformulating the task as independent binary classification tasks. Thus, CAVs and concept importances were learned for each task independently.

C.6. CRAFT

The open-source implementation provided in the Xplique toolbox [11] (by the same authors) was used to evaluate this approach. CRAFT uses NMF to find CAVs and introduces a Sobol-based sensitivity approach to find concept importance. Embeddings are projected to the concept space using the toolbox's built-in transform method.

C.7. ConceptSHAP

ConceptSHAP differs from previous U-CBEMs because it discovers CAVs globally (i.e., class-agnostic). Thus, we need a way to associate discovered CAVs to the concepts. To do so, we discover a (larger than K) set of CAVs and assign them a per-class concept importance score (through Shapley Values). Then, the top K CAVs by importance magnitude are selected to be the CAVs for a given class. For all experiments, we initially discover 100 CAVs. This number is arbitrary, but note that the Shapley Value computation scales exponentially with the number of initially discovered CAVs, which motivated our choice. With more CAVs, we expect ConceptSHAP to obtain higher faithfulness at the expense of a higher computational cost. Embeddings are linearly projected to the concept space. For age regression, we only discover K concepts (since there are no classes). For multi-attribute prediction, we repeat the ConceptSHAP procedure for each attribute, considering each attribute prediction as a binary classification task; we only discover K concepts, since each task is a binary classification.

The authors provide an open-source implementation in TensorFlow, so we instead use a third-party re-implementation in PyTorch. Several modifications are made to this implementation:

1. Added in a learnable two-layer MLP to map back to the representation space (as done in the original paper)
2. Added more efficient concept importance scoring via Shapley Values. This is implemented using KernelSHAP ([33]) and is class-specific.
3. Fixed the implementation of the completeness score

C.8. MCD

Instead of representing concepts as single CAVs, MCD represents concepts as multi-dimensional subspaces. In [44], any concept l for class i is characterized by a representative basis $C^{i,l} = \{\mathbf{v}_j^{i,l} | j = 1...d_l\}$, where d_l denotes the dimensionality of the subspace. To ensure the union of all concepts span the entire feature space, they define the orthogonal complement concept $C^{i,\perp} = \text{span}(C^{i,1}, ..., C^{i,K})^{\perp}$, where K is the number of concepts originally discovered, letting $C^{i,K+1} \equiv C^{i,\perp}$. Then, any embedding $\mathbf{h}$ can be uniquely decomposed onto the concept basis as:

$$\mathbf{h} = \sum_{l=1}^{n_c+1} \sum_{j=1}^{d^l} p_j^{i,l} \, \mathbf{v}_j^{i,l} \tag{15}$$

Thus, we stack all $p_j^{i,l}$ into the concept representation $\mathbf{p_i}$ for class i. Since $C^{i,\perp}$ is not interpretable, we set elements of $\mathbf{p_i}$ corresponding to $C^{i,\perp}$ to 0; thus, the reconstruction of $\mathbf{h}$ is done using interpretable concepts presented to the user during an explanation. We set concept importance $\alpha_j^{i,l} = \mathbf{f}_i^T \mathbf{v}_j^{i,l}$, which directly relates to the local and global concept relevance defined in MCD and allows for faithful reconstruction of the output using our surrogate.

Since MCD is computationally expensive when trying to discover concepts on a large number of embeddings, we randomly sample 10,000 embeddings if there are greater than these many embeddings extracted for a given class. We use the open-source implementation provided by the authors.

Table 5. **Additional Benchmark Results.** We apply SURF to evaluate explanations from prior U-CBEMs on the *Object Classification* task from Sec. 6.2 on a VGG11 and InceptionV3. Along with the SURF_{MAE} and SURF_{EMD}, we report other metrics to serve as a comparison. Our trends hold: prior U-CBEMs are not faithful, as indicated by large errors in the logit and probability space.

(a) Object classification (VGG11)

U-CBEM	SURF_{MAE} (↓)	SURF_{EMD} (↓)	Top-1 (%) (↑)	Rank Corr (↑)
CDISCO	2.89	0.936	0.4	-0.01
ICE	3.58	0.616	**93.7**	0.498
CRAFT	2.50	0.844	51.0	0.214
C-SHAP	3.28	0.884	1.1	-0.00
MCD	1.95	0.738	83.3	0.306
HU-MCD	–	–	–	–
SAE	**0.47**	**0.567**	62.4	**0.504**

(b) Object classification (InceptionV3)

SURF_{MAE}	SURF_{EMD}	Top-1 (%)	Rank Corr
4.08	0.847	1.52	0.012
3.34	0.595	50.3	-0.05
4.26	0.652	8.41	0.131
4.31	0.754	9.70	-0.08
2.76	0.327	**89.1**	0.385
–	–	–	–
0.35	**0.226**	72.0	**0.770**

C.9. HU-MCD

HU-MCD directly extends upon MCD, adopting the same framework of concept subspace, orthogonal complement, and unique decomposition onto the concept basis. The only material difference is in the concept discovery stage, where concepts are discovered on representations arising from image segments, instead of from the feature map. Therefore, we make the same adaptations as MCD for HU-MCD. We use the open-source implementation provided by the authors.

C.10. SAE

We learn a Top-K SAE [16] on the training embeddings (before average pooling) using the Overcomplete library. SAEs learn class-agnostic concepts (in contrast to U-CBEMs that learn CAVs for a specific class output); to enable a fair comparison, our SAE learns the same number of total CAVs as the other U-CBEMs for each setting. As an example, our SAE for the *Object Classification* task learns 505 CAVs (5 per class output, 101 total classes). At reconstruction time, we set the K in the Top-K SAE to 5, directly mirroring the other U-CBEMs.

D. Additional Benchmark Results

We repeat our experiment on the *Object Classification* task from Sec. 6.2 on a VGG11 and InceptionV3 to confirm trends hold. We use the ImageNet pre-trained weights, and finetune them for the *Object Classification* task. Results are reported in Tab. 5. For both models, SAE is the most faithful. For VGG11, we find that all U-CBEMs tend to exhibit large EMD errors (larger than 0.5). Though SAE is able to reduce the logit error significantly, the metrics indicate that there are still issues when trying to reproduce predictions in the probability space. While ICE can accurately reconstruct the Top-1 prediction, it has large logit errors. For InceptionV3, we find ICE is not faithful, and instead MCD is second-best (after SAE). Compared to the ResNet studied in Sec. 6.2, we believe that the models tested here have more nuance in their predicted probability space, which may lead to larger EMD errors, even if the logit error seems small.