POST-HOC SELF-EXPLANATION OF CNNS

Anonymous authors
Paper under double-blind review

ABSTRACT

Although standard Convolutional Neural Networks (CNNs) can be mathematically reinterpreted as Self-Explainable Models (SEMs), their built-in prototypes do not on their own accurately represent the data. Replacing the final linear layer with a k-means-based classifier addresses this limitation without compromising performance. This work introduces a common formalization of k-means-based post-hoc explanations for the classifier, the encoder's final output (B4), and combinations of intermediate feature activations. The latter approach leverages the spatial consistency of convolutional receptive fields to generate concept-based explanation maps, which are supported by gradient-free feature attribution maps. Empirical evaluation with a ResNet34 shows that using shallower, less compressed feature activations, such as those from the last three blocks (B234), results in a trade-off between semantic fidelity and a slight reduction in predictive performance.

1 INTRODUCTION

Convolutional Neural Networks (CNNs) are the basis for several foundation models, particularly for image classification Deng et al. (2009), but are also utilized for other data types Ribeiro et al. (2020). Their architectures, inspired by the biological visual cortex, typically consist of a feature extractor or encoder that combines 2D convolutions, activation functions, dropout, and normalization layers, followed by an average pooling and finally a linear classifier or regressor. This structure is present in many modern models, including ResNet He et al. (2016) and DenseNet Huang et al. (2017). In some cases, a dropout layer precedes the classifier, as in GoogleNet Szegedy et al. (2015) or EfficientNet Tan & Le (2019). Some variants use a multilayer perceptron as the classifier Krizhevsky et al. (2012); Simonyan & Zisserman (2014). The consistent element is the average pooling between the encoder and the classifier. This paper shows that this key operator renders these models structurally and mechanistically analogous to self-explainable models.

Self-explainable models (SEMs) constitute a class of architectures designed to improve the interpretability and transparency of deep learning models by explicitly associating predictions with human-understandable concepts or prototypes. Here, we distinguish these two by defining concepts as frequent/relevant patterns in the training dataset that may possess varying degrees of semantic meaning, such as ”a blue sky” or ”a plane.” Prototypes are vectors representing these concepts in the embedding space, and the similarity between input data and these prototypes provides concept-based explainability for predictions. The selection of concepts and prototypes is a critical consideration. Prototypes selected a priori Kim et al. (2018) may introduce biases and fail to capture the full diversity of the dataset Celis et al. (2016). Jointly learning prototypes during training comes at a cost of a complicated alternating training scheme and may negatively affect the performance Alvarez Melis & Jaakkola (2018); Chen et al. (2019); Kjærsgaard et al. (2024). Additionally, because these architectures differ from the traditional classifiers they are intended to explain or replace, the interpretability insights obtained may not transfer seamlessly.

This work advocates for computing prototypes after training. Indeed, the classic feedforward architecture remains highly versatile and efficient. A common formalization framework for k-means-based post-hoc explanations is introduced, covering three explanation locations: the classifier weights, the encoder's final output, and its intermediate feature activations. It encompasses the cases where the CNN is reinterpreted as a SEM, the linear classifier is substituted with a k-means-based one as described in Gautam et al. (2024), and the multi-depth explanation of the encoder building Boubekki et al. (2025). The latter is here further extended to produce gradient-free and concept-aligned feature attribution maps.

2 Problem Definition and Notations

Notational conventions follow Goodfellow et al. (2016): capital letters denote dimensions, dot products are written $\cdot$, and biases are omitted from classifiers without loss of generality. The classifier is represented by a matrix $\boldsymbol{C}$ with columns $\boldsymbol{c}_j$.

A standard CNN-based classifier architecture comprises a CNN encoder $\mathrm{enc} : \mathbb{R}^Q \to \mathbb{R}^{R\times D}$, an average pooling operation $\mathrm{avg.\,pool} : \mathbb{R}^{R\times D} \to \mathbb{R}^D$, and a linear classifier $\mathrm{clf} : \mathbb{R}^D \to \mathbb{R}^C$. The encoder output is typically a tensor of dimension (W, H, D). However, for legibility, the first two dimensions are here flattened. The operations are formally defined as follows:

$$\boldsymbol{x} \xmapsto{\mathrm{enc}} \boldsymbol{h} \xmapsto{\mathrm{avg.pool}} \boldsymbol{z} \xmapsto{\mathrm{clf}} \boldsymbol{y}, \tag{1}$$

where $\boldsymbol{x} \in \mathbb{R}^Q$, $\boldsymbol{h} \in \mathbb{R}^{R\times D}$, $\boldsymbol{z} \in \mathbb{R}^D$ and $\boldsymbol{y} \in \mathbb{R}^C$, and:

$$\mathrm{avg.\,pool}(\boldsymbol{h}) = \frac{1}{R}\sum_{r=1}^{R} \boldsymbol{h}_r = \boldsymbol{z} \quad \text{and} \quad \mathrm{clf}(\boldsymbol{z}) = \boldsymbol{z}\cdot\boldsymbol{C} = (\boldsymbol{z}\cdot\boldsymbol{c}_j)_{j=1\ldots C}. \tag{2}$$

SEM Self-explainable models differ from classic CNNs by their output layer. In this case, a classifier which compares a feature vector, also noted $\boldsymbol{h}$, to a set of $K > 0$ prototypes $\boldsymbol{P} = \{\boldsymbol{p}_k\}_K$ using a similarity measure sim. The similarity scores are then mapped using proj into a prediction score $\boldsymbol{y} \in \mathbb{R}^C$.

$$\boldsymbol{x} \xmapsto{\mathrm{enc}} \boldsymbol{h} \xmapsto{\mathrm{sim}} \Big(\mathrm{sim}(\boldsymbol{h}_r, \boldsymbol{p}_k)\Big)_{R,K} \xmapsto{\mathrm{proj}} \boldsymbol{y}. \tag{3}$$

The dimensions of $\boldsymbol{h}$ are intentionally omitted as the location of the prototypical classifier depends on which part of the CNN we want to explain. The similarity measure varies between models and is usually implemented as either a dot product Parekh et al. (2021) or a distance metric Chen et al. (2019). In order to remain *transparent*, proj should remain as simple as possible. Typically, it is either a matrix multiplication Chen et al. (2019) or a pooling operation.

Explaining with ProtoPNet We use ProtoPNet introduced in Chen et al. (2019) as the SEM baseline. It employs an alternating update strategy: the encoder is updated via gradient descent, and the prototype layer is updated via a multi-stage procedure. The similarity measure is ℓ_2-based, and proj is a linear layer.

3 Post-hoc Self-explanation of CNNs

This section presents a common formalization for three k-means-based post-hoc self-explanation of frozen CNNs.

3.1 Self-explaining the Classifier

A CNN can be reinterpreted as a SEM. Indeed, since the classifier is a linear layer, its column vectors can be viewed as prototypes, and the matrix operation as the similarity measure.

Theorem 1. *Convolutional neural network classifiers are self-explainable models with C prototypes corresponding to the vector columns of the classifier.*

Proof. Consider a CNN classifier as defined in Equations 1 and 2. The commutativity of the average pool and dot-product operations means that the final prediction is also the average prediction of each $\boldsymbol{h}_r$:

$$\mathrm{clf}\circ\mathrm{avg.\,pool}(\boldsymbol{h}) = \left(\left(\frac{1}{R}\sum_{r=1}^{R}\boldsymbol{h}_r\right)\cdot\boldsymbol{c}_j\right)_{j=1\ldots C} = \left(\frac{1}{R}\sum_{r=1}^{R}(\boldsymbol{h}_r\cdot\boldsymbol{c}_j)\right)_{j=1\ldots C} \tag{4}$$

By setting $\mathrm{proj} = \mathrm{avg.\,pool}$, defining sim as the dot-product, and treating $\boldsymbol{c}_j$ as prototypes, the following is obtained:

$$\mathrm{clf} \circ \mathrm{avg.\,pool} \circ \mathrm{enc}(\boldsymbol{x}) = \frac{1}{R} \sum_R (\boldsymbol{h}_r \cdot \boldsymbol{c}_j) = \mathrm{proj} \circ \mathrm{sim} \circ \mathrm{enc}(\boldsymbol{x}) \tag{5}$$

Therefore, the operations of the CNN can be reinterpreted as those of an SEM, as defined by Equation 3. □

In practice, the $\boldsymbol{c}_j$ vectors are not satisfactory prototypes, particularly with respect to diversity. Their number is inherently limited by the number of classes. Preliminary experiments indicate that simply increasing the number of classes and merging them using max or average pooling, without additional regularization, results in a single direction dominating each class. Although the cross-entropy loss encourages prediction separability, leading to class embeddings covering distinct half-spaces, it does not guarantee that points are centered or distributed along the line defined by the prototype or column vector, meaning that prototypes may be far from the data. This contradicts the implicit requirement of representativeness or diversity of the prototypes.

3.2 Explaining the Classifier

A workaround to the lack of representativeness is to replace the classifier with a k-means based one as introduced in Gautam et al. (2024). As the centroids of class-wise clusterings, the prototypes are thus more likely to be closer to the data. The procedure is as follows:

1. For each of the C classes, K/C prototypes are learned using k-means on $\boldsymbol{z}$ of the training data.
2. The similarity measure is the exponential of minus the ℓ_2 distance: $\mathrm{sim}(\boldsymbol{h}, \boldsymbol{p}_k) = \exp(\text{-}||\boldsymbol{h}, \boldsymbol{p}_k||^2)$.
3. The proj returns a one-hot vector centered on the class of the prototype with largest similarity score.

The original KMEx utilizes the ℓ_2 distance as a similarity measure and employs a nearest neighbor classifier. The similarity described in the second step is equivalent and allows to derive the class predictions using an $\arg\max$. Although the method can accommodate varying numbers of prototypes per class, we fix them to K/C prototypes per class.

3.3 Explaining the Encoder

To explain the encoder rather than the classifier, intermediate outputs, also referred to as feature activations, are compared to prototypes instead of the embedding vector after average pooling, like in the previous section. This approach aims to access information that may be filtered out before reaching the classifier and provide explanations at the patch or segment level. ProtoPNet is an example of such an approach, as its so-called *part* prototypes are compared to the pixels of the encoder's output. Other models Parekh et al. (2021); Zhu et al. (2025) utilize shallower feature activations to compute part prototypes. In these cases, the prototypes are composite representations that integrate information from multiple depths. The post-hoc framework presented here follows the rationale of KMEx Gautam et al. (2024) by using k-means to learn prototypes on a frozen backbone. Formally, it extends the work in Boubekki et al. (2025) to compute predictions and feature attributions.

Extracting Feature Activations Let us first decompose the encoder into B blocks of layers:

$$\boldsymbol{x} \overset{f_1}{\longmapsto} \boldsymbol{x}^{(1)} \cdots \overset{f_b}{\longmapsto} \boldsymbol{x}^{(b)} \cdots \overset{f_B}{\longmapsto} \boldsymbol{x}^{(B)} = \boldsymbol{h}, \quad \text{where} \quad \boldsymbol{x}^{(b)} \in \mathbb{R}^{R_b \times D_b}. \tag{6}$$

Since the outputs of each block have different resolutions R_b and numbers of channels or dimensions D_b, preprocessing steps are required to compute a composite matrix $\check{\boldsymbol{h}}$. This matrix is then compared to the prototypes to generate a prediction vector $\boldsymbol{y}$. The K/C class prototypes are learned using k-means on the row vectors $\check{\boldsymbol{h}}_r$ of the class data. The operations are as follows.

1. The intermediate outputs are first linearly interpolated to a shared resolution R'.

$$\mathrm{Upsample}\left(\boldsymbol{x}^{(b)}\right) = \boldsymbol{u}^{(b)} \in \mathbb{R}^{R' \times D_b}. \tag{7}$$

2. The $\boldsymbol{u}^{(b)}$ are then normalized and scaled before being concatenated into $\check{\boldsymbol{h}}$

$$\mathrm{Concatenate}\left(\frac{\boldsymbol{u}^{(b)}}{D_b ||\boldsymbol{u}^{(b)}||}\right) = \check{\boldsymbol{h}} \in \mathbb{R}^{R' \times (\sum_b D_b)} \tag{8}$$

3. The assignments of the rows $\breve{\boldsymbol{h}}_r$ to the closest prototypes are stored in a *binary matrix*:

$$\mathrm{sim}\left((\boldsymbol{x}^{(b)})_B, \boldsymbol{P}\right) = \mathrm{sim}\left(\breve{\boldsymbol{h}}, \boldsymbol{P}\right) = \arg\min_k(||\breve{\boldsymbol{h}}_r - \boldsymbol{p}_k||) = \breve{\boldsymbol{s}} \in \mathbb{R}^{R' \times K} \tag{9}$$

4. The prediction vector is computed as the average count of the class-wise clusters in $\breve{\boldsymbol{s}}$, and the predicted class corresponds to the most frequently occurring cluster.

$$\mathrm{proj}(\breve{\boldsymbol{s}}) = \mathrm{avg.\,pool}\left(\sum_r \breve{\boldsymbol{s}}_r \, ; K/C\right) = \boldsymbol{y} \in \mathbb{R}^C \tag{10}$$

The average pooling operation, which uses K/C non-overlapping windows, assumes a constant and ordered number of clusters per class. The assignment binary matrix $\breve{\boldsymbol{s}}$ can be interpreted as a low-resolution segmentation of the input, referred to as an "*explanation map*". In practice, outputs from all blocks after a specified depth are combined. Accordingly, the notations $\breve{\boldsymbol{h}}^{(b:)}$ and $\breve{\boldsymbol{s}}^{(b:)}$ indicate that all outputs from f_b to f_B are utilized.

Feature Importance Numerous feature attribution methods have been proposed Sundararajan et al. (2017); Selvaraju et al. (2017); Petsiuk et al. (2018), resulting in a wide variety of outputs. The present work introduces a simplified approach that leverages the self-explainable reinterpretation of a CNN and the commutativity of average pooling with respect to the dot product. To keep track of the input format, this section explicitly indicates the width W_b and height H_b of the outputs, rather than the aggregated R_b dimension.

Analogous to the class activation map (CAM) Zhou et al. (2016), the feature attribution score of pixel $\boldsymbol{h}_{wh}$ is defined as a measure of its alignment with $\boldsymbol{c}_j$.

$$\mathrm{att}(\boldsymbol{h}_{wh}, \boldsymbol{c}_j) = \frac{\boldsymbol{h}_{wh} \cdot \boldsymbol{c}_j}{||\boldsymbol{c}_j||^2} \tag{11}$$

The distinction lies in the normalization by the squared norm, which ensures that $\mathrm{att}(\boldsymbol{c}_j^T, \boldsymbol{c}_j) = 1$ and allows values to be compared across classes.

The resulting attribution map matches the low resolution of the encoder's output. Rather than relying on assumptions about the information conveyed by gradients, as in Grad-CAM variants Selvaraju et al. (2017), the upstream feature attribution map is approximated by exploiting the spatial consistency of the convolutional receptive fields. Namely, the attribution map at depth b is computed from the one at depth $b + 1$ as follows:

1. Compute the explanation map at depth b, $\breve{\boldsymbol{s}}^{(b:)} \in \mathbb{R}^{W_b \times H_b \times K}$.
2. Upsample $\mathrm{att}^{(b+1:)} \in \mathbb{R}^{W_{b+1} \times H_{b+1}}$ to $\mathrm{att}^{(b+1:,\mathrm{up})} \in \mathbb{R}^{W_b \times H_b}$.
3. Compute $\mathrm{att}^{(b:)}$ as the segment-wise average of $\mathrm{att}^{(b+1:,\mathrm{up})}$ based on $\breve{\boldsymbol{s}}^{(b:)}$.

The segment-wise averaging results in discrete attribution, where pixels within the same segment share an identical attribution score.

Interpretation Process Figure 1 illustrates the interpretation process of the k-means based explanation of the encoder. Each input receives two forms of explanation: a segmentation map and a feature importance map. The segmentation map delineates regions of the input that yield similar feature activations, resulting in clustered segments. Cluster interpretation is supported by visualization of the nearest segment to associated prototype, using a consistent color scheme. The feature importance map highlights the relevance of each concept with respect to the classifier of the backbone model. Incorporating shallower feature activations enhances the resolution and detail of the map.

4 EXPERIMENTS

Experimental Setting All experiments use a ResNet34 backbone pretrained on ImageNet, trained following Gautam et al. (2024), and evaluated on MNIST (Lecun et al. (1998)), STL10 (Coates et al. (2011)), and CUB-200 (Wah et al. (2011)), with five prototypes per class for the first two and ten

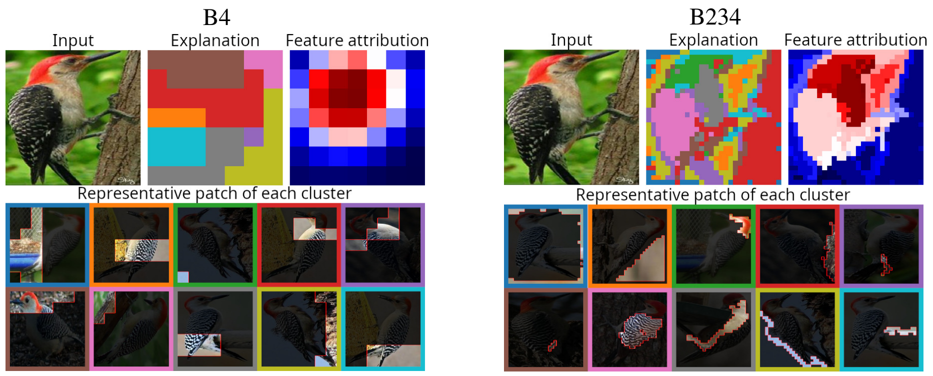

Figure 1: Interpretation process for B4 (left) and B234 (right) on a CUB-200 red-bellied woodpecker. Red and blue indicate higher and lower feature importance. Representative patches are the closest training examples to each prototype; border colors match the explanation map segments.

for CUB-200, if not specified otherwise. Our method is referenced according to the depth of the feature activations employed. The ResNet34 backbone consists of a series of preprocessing layers and four residual blocks. The model utilizing the same information as ProtoPNet, specifically at the encoder's output (the fourth block), is designated as B4. The model B234, which incorporates outputs from the last three blocks, is also evaluated. We compare these to KMEx and ProtoPNet.

Alignment of the Prototypes The column vectors $\mathbf{c}_j$ of the classifier define the half-space in which the class embedding predominantly resides, rather than the direction along which the data spreads. Table 1 validates this interpretation using the average cosine similarity between each method's prototypes and their class embedding (*class*) and all other data points (*out*).

The uniformly high cosines of ProtoPNet reflect the distinct embedding geometry produced by an integrated training. The *class* cosines of the CNN's classifier weights are approximately 0.5, versus nearly 1 for KMEx, indicating poor alignment of the $\mathbf{c}_j$ with the data. The *out* cosines suggest that the $\mathbf{c}_j$ define nearly orthogonal half-spaces, though the class embeddings themselves are not fully orthogonal. Figure 2 illustrates this misalignment via a UMAP projection McInnes et al. (2018) of the CNN's embeddings for the twenty sparrow classes of CUB-200. The classifier prototypes $\mathbf{c}_j$, both raw (crosses) and rescaled to class norm (triangles), are dispersed and rarely near their respective class; by contrast, the KMEx prototypes (squares) lie within their respective clusters.

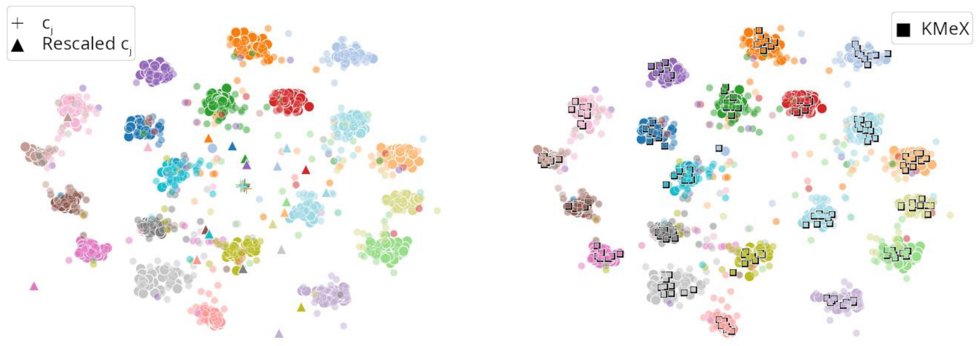

Figure 2: UMAP projection of training and test (lower opacity) embeddings from the twenty sparrow classes of CUB-200. Left: classifier prototypes $\mathbf{c}_j$ as crosses and rescaled to class norm as triangles. Right: KMEx prototypes.

Concept-based Accuracy Regarding train and test accuracies (Table 2), replacing the linear classifier with KMEx does not affect the performance. This outcome is expected, since the prototypes of KMEx are learned class-wise and based on the same information as the $\mathbf{c}_j$. Although B4 derives

Table 1: Average cosine of classifier prototypes within *class* and *out* of class embedding.

	MNIST		STL10		CUB200	
Model	class	out	class	out	class	out
CNN	0.48	-0.05	0.46	-0.04	0.51	0.00
KMEx (K/C=1)	0.88	0.32	0.79	0.42	0.93	0.47
KMEx (K/C=5)	0.83	0.30	0.84	0.34	0.87	0.45
ProtoPNet	1.00	1.00	0.95	0.92	0.98	0.97

Table 2: Average train and test accuracies over five runs. Models were pretrained on ImageNet.

	MNIST		STL10		CUB200	
Model	train	test	train	test	train	test
CNN	100.0	99.4	95.6	86.1	100.0	79.1
KMEx (K/C=5)	100.0	99.4	94.9	85.6	100.0	78.6
ProtoPNet	100.0	99.4	74.2	72.7	99.4	66.4
ProtoPNet in Chen et al. (2019)						79.2
Ours B4	100.0	99.4	94.0	85.5	100.0	75.2
Ours B234	98.9	98.5	83.5	77.9	85.5	52.8

the class predictions from the concept proportions, it matches the performance of the backbone network (Table 2). However, including information at shallower depths, as in B234, renders the classification signal less distinct, leading to reduced performance.

5 Discussion and Related Work

Concept-based explainability methods for deep networks have been extensively reviewed in Lee et al. (2025). The proposed framework offers a common formalization of k-means-based post-hoc explanations, explicitly avoiding gradient- or perturbation-based techniques. We position our work with respect to the most closely related methods for explaining the encoder. TCAV (Kim et al. (2018)) probes feature activations at a single depth using user-defined concept vectors and assigns feature importance using directional derivatives of the classifier's output with respect to the concept vectors. Its extension, ACE (Ghorbani et al. (2019)), automates concept discovery by segmenting inputs with SLIC (Achanta et al. (2012)) and processing the resulting crops through the encoder. Since SLIC segmentation operates independently of the encoder's learned representations, it explains image regions rather than internal concepts. Additionally, projecting crops through an encoder trained on complex images is likely to result in out-of-distribution behavior. Substituting TCAV with SHAP produces CONE-SHAP (Li et al. (2021))), which retains the same limitations. Network Dissection (Bau et al. (2017)) and Net2Vec (Fong & Vedaldi (2018)) probe individual neurons at a single depth to identify human-interpretable concepts. For the final layer of a ResNet34 (B4), this involves comparing 512 neurons, which, as reported by the authors, are often redundant. CRAFT (Fel et al. (2023)) and ICACE (Zhang et al. (2021)) apply matrix factorization to feature activations to extract concepts, and leverage implicit differentiation and non-negative constraints, respectively, to produce concept attribution maps. However, restricting the back-propagated signal to a single class leaves portions of the input unexplained.

6 Conclusion

This paper introduces a common formalization of k-means-based post-hoc explanations covering three locations: the classifier weights, the encoder's final output, and its intermediate feature activations. The commutativity of average pooling and the linear classifier shows that CNNs are mechanistically analogous to prototype-based SEMs. However, the classifier's column vectors, when interpreted as prototypes, are poorly aligned with the data, undermining their representativeness. Replacing them with k-means centroids addresses this limitation without loss of performance. Extending this to intermediate feature activations produces detailed, dense explanation maps supported by semantically relevant prototypes. Attribution maps are computed via a gradient-free variant of CAM that leverages the spatial consistency of convolutional receptive fields. Together, these results suggest that the perceived black-box nature of CNNs is not an inherent property, but one that can be mitigated through rigorous reinterpretation of their existing operations.

The primary limitation is the scalability of k-means to large datasets, though preliminary results suggest that performance remains stable on subsets of the training data. The sustained classification performance of B4 and B234 using cluster distributions calls for further investigation.

REFERENCES

Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2274–2282, 2012.

David Alvarez Melis and Tommi Jaakkola. Towards robust interpretability with self-explaining neural networks. *Advances in neural information processing systems*, 31, 2018.

David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6541–6549, 2017.

Ahcène Boubekki, Samuel G. Fadel, and Sebastian Mair. Explaining the Impact of Training on Vision Models via Activation Clustering, March 2025. URL `http://arxiv.org/abs/2411.19700`. arXiv:2411.19700 [cs].

L Elisa Celis, Amit Deshpande, Tarun Kathuria, and Nisheeth K Vishnoi. How to be fair and diverse? *arXiv preprint arXiv:1610.07183*, 2016.

Chaofan Chen, Oscar Li, Daniel Tao, Alina Barnett, Cynthia Rudin, and Jonathan K Su. This looks like that: deep learning for interpretable image recognition. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In Geoffrey Gordon, David Dunson, and Miroslav Dudík (eds.), *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pp. 215–223, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009.

Thomas Fel, Agustin Picard, Louis Bethune, Thibaut Boissin, David Vigouroux, Julien Colin, Rémi Cadène, and Thomas Serre. Craft: Concept recursive activation factorization for explainability. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2711–2721, 2023.

Ruth Fong and Andrea Vedaldi. Net2vec: Quantifying and explaining how concepts are encoded by filters in deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8730–8738, 2018.

Srishti Gautam, Ahcene Boubekki, Marina MC Höhne, and Michael Kampffmeyer. Prototypical self-explainable models without re-training. *Transactions on Machine Learning Research*, 2024.

Amirata Ghorbani, James Wexler, James Y Zou, and Been Kim. Towards automatic concept-based explanations. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016. URL `http://arxiv.org/abs/1512.03385`. arXiv: 1512.03385.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, et al. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In *International conference on machine learning*, pp. 2668–2677. PMLR, 2018.

Rune Kjærsgaard, Ahcène Boubekki, and Line Clemmensen. Pantypes: Diverse representatives for self-explainable models, 2024. URL `https://arxiv.org/abs/2403.09383`.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pp. 1097–1105, 2012.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Jae Hee Lee, Georgii Mikriukov, Gesina Schwalbe, Stefan Wermter, and Diedrich Wolter. Concept-based explanations in computer vision: Where are we and where could we go? In *Computer Vision – ECCV 2024 Workshops*, volume 15643 of *Lecture Notes in Computer Science*, pp. 266–287. Springer Nature Switzerland, 2025.

Jiahui Li, Kun Kuang, Lin Li, Long Chen, Songyang Zhang, Jian Shao, and Jun Xiao. Instance-wise or class-wise? a tale of neighbor shapley for concept-based explanation. In *Proceedings of the 29th ACM International Conference on Multimedia*, pp. 3664–3672, 2021.

Leland McInnes, John Healy, Nathaniel Saul, and Lukas Großberger. Umap: Uniform manifold approximation and projection. *Journal of Open Source Software*, 3(29):861, 2018.

Jayneel Parekh, Pavlo Mozharovskyi, and Florence d'Alché Buc. A framework to learn with interpretation. *Advances in Neural Information Processing Systems*, 34:24273–24285, 2021.

Vitali Petsiuk, Abir Das, and Kate Saenko. RISE: Randomized input sampling for explanation of black-box models. In *British Machine Vision Conference*, 2018.

Antonio H Ribeiro, Daniel Gedon, Daniel Martins Teixeira, Manoel Horta Ribeiro, Antonio L Pinho Ribeiro, Thomas B Schon, and Wagner Meira Jr. Automatic 12-lead ECG classification using a convolutional network ensemble. In *Computing in Cardiology (CinC)*, 2020. doi: 10.22489/CinC.2020.130.

Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-CAM: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 618–626, 2017.

Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv:1409.1556 [cs]*, September 2014. URL `http://arxiv.org/abs/1409.1556`. 00000 arXiv: 1409.1556.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International Conference on Machine Learning*, pp. 3319–3328. PMLR, 2017.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pp. 6105–6114. PMLR, 2019.

C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The caltech-ucsd birds-200-2011 dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

Ruihan Zhang, Prashan Madumal, Tim Miller, Krista A. Ehinger, and Benjamin I. P. Rubinstein. Invertible concept-based explanations for CNN models with non-negative concept activation vectors. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 11682–11690, 2021.

Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2921–2929, 2016.

Zhijie Zhu, Lei Fan, Maurice Pagnucco, and Yang Song. Interpretable image classification via non-parametric part prototype learning. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 9762–9771, 2025.