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 fail to accurately represent the data. This limitation is addressed by introducing a post-hoc framework that substitutes these internal weights with prototypes learned by k-means on feature activations. By utilizing shallower and thus less compressed feature activations, the proposed method provides detailed explanations of the predictions in the form of a segmentation map supported by a gradient-free attribution maps. The results demonstrate a trade-off of the method: employing deep feature activations (B4) enables the model to maintain its original accuracy, whereas incorporating earlier layers (B234) yields sharper and more interpretable explanation maps, at a slight cost to 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). 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. It is thus reasonable to explore methods to extract relevant information from it and process it like an SEM. Several methods of increasing complexity are presented, beginning with the direct interpretation of the CNN as an SEM. After identifying the limitations of the built-in prototypes, alternative approaches to learning prototypes using k-means Gautam et al. (2024) are explored. Ultimately, a novel method inspired by Boubekki et al. (2025) is introduced, which utilizes feature activations from the encoder to generate both an explanation map in the form of a segmentation and a feature attribution map with respect to the

backbone's classifier. Involving shallower intermediate outputs tends to decrease accuracy but yields more detailed explanation maps and more diverse concepts.

2 BACKGROUND

Notations The notational conventions are adopted from Goodfellow et al. (2016), with a few exceptions. Capital letters are used to denote dimensions, thereby enhancing traceability and clearly distinguishing them from scalars. Index ranges in summations, and vector or matrix notations are omitted when context allows. The encoder output is typically represented as a vector, although it may be a matrix depending on the context. Classifiers are, without loss of generality, assumed to exclude biases and are represented by a matrix $\boldsymbol{C}$ with columns denoted as $\boldsymbol{c}_j$. Finally, the dot product is noted $\cdot$.

CNN 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:

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

SEM Self-explainable models differs from CNNs by their classifier which compares a feature vectors, also noted $\boldsymbol{h}$, to a set of $K > 0$ prototypes $\boldsymbol{P} = \{\boldsymbol{p}_k\}_K$ using a similarity measure sim. The similarity score 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.

Experimental Setting To ensure consistency in the experimental setup, the ResNet34 is the single backbone is used. The training procedure adheres to the methodology outlined in Gautam et al. (2024). Three datasets of increasing complexity are employed: MNIST Lecun et al. (1998), STL10 Coates et al. (2011), and CUB-200 Wah et al. (2011). Five prototypes per class are learned for MNIST and STL10, and ten prototypes per class are learned for CUB-200.

3 EXPLAINING THE CLASSIFIER

Reinterpreting the CNN A CNN can be reinterpreted as an 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 the final predictions is also the average prediction of each $\boldsymbol{h}_r$:

$$\mathrm{clf} \circ \mathrm{avg.\,pool}(\boldsymbol{h}) = \big((\frac{1}{R}\sum_{R} h_r)\cdot\boldsymbol{c}_j\big)_C = \frac{1}{R}\sum_{R}(h_r\cdot\boldsymbol{c}_j) \tag{4}$$

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

Model	MNIST class	MNIST out	STL10 class	STL10 out	CUB200 class	CUB200 out
CNN	0.48	-0.05	0.46	-0.04	0.51	0.00
KMEx (K/C=5)	0.83	0.30	0.84	0.34	0.87	0.45
ProtoPNet (ours)	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.

Model	MNIST train	MNIST test	STL10 train	STL10 test	CUB200 train	CUB200 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 (ours)	100.0	99.4	74.2	72.7	99.4	66.4
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

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

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

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 satisfying 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, causing class embeddings to occupy 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 which contradicts the implicit requirement of representativeness or diversity of the prototypes.

Explaining with KMEx The k-means explainer (KMEx), introduced in Gautam et al. (2024), replaces the classifier of a trained and frozen CNN with a prototypical classifier. In this approach, the prototypes are centroids derived from multiple class-wise k-means clusterings. Using the notation from Equation 2, 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 similarity scores also serves as prediction scores, i.e., proj is the identity.

The original KMEx utilizes the ℓ_2 distance as a similarity measure and employs a nearest neighbor classifier. In this work, equivalent operations are used so that predictions are based on the largest value. Although the method can accommodate varying numbers of prototypes per class, we fix them to K/C prototypes per class.

Alignment If the column vectors $\boldsymbol{c}_j$ of the classifier are interpreted as prototypes, they define the half-space in which the class embedding predominantly resides, rather than the direction along which it spreads. To validate this interpretation, the average cosine similarity between each prototype and its class embedding, as well as with all other data points, is computed (Table 1). For comparison, prototypes learned using KMEx and ProtoPNet are also included.

The consistency of the *out* cosines of the CNN around zero indicates that cross-entropy loss seeks a balance between aligning $\boldsymbol{c}_j$ with their respective class and maintaining orthogonality with others. The *out* cosines reported by KMEx demonstrate that class embeddings are not fully orthogonal. Additionally, the large *class* cosines confirm that $\boldsymbol{c}_j$ are not aligned with the data. Notably, the prototypes computed by ProtoPNet generate directions that are perfectly aligned with the data both within and outside their class. This observation suggests that the distance-based similarity employed during training produces a distinct type of embedding.

Accuracy Regarding train and test accuracies (Table 2), replacing the linear classifier with KMEx does not reduce performance. This outcome is expected, as the prototypes are learned in a class-wise manner. Increasing the number of prototypes further refines the partitioning of the embedded space, which should result in improved test accuracy.

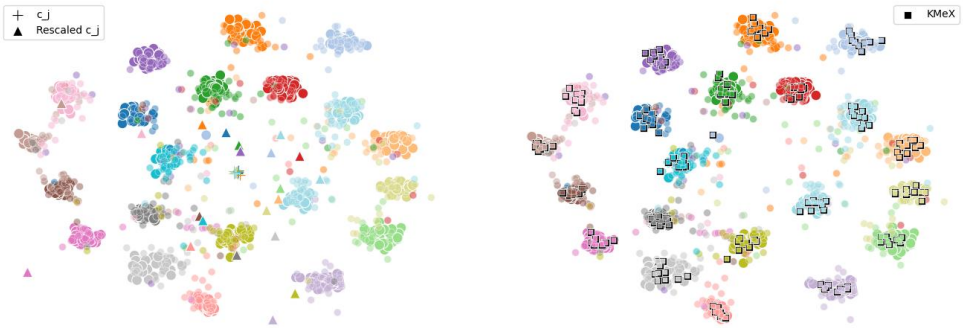

Figure 1: A UMAP projection of the training and test (with lower opacity) samples from the twenty sparrow classes of CUB-200 is presented. On the left, crosses represent the classifier prototypes $\boldsymbol{c}_j$. Due to their small norms, these prototypes are also plotted as triangles scaled to the average norm of their respective class. On the right, the same UMAP projection is shown with the KMEx prototypes.

Embedding Figure 1 presents a UMAP projection McInnes et al. (2018) of the embedding vectors for training and test images from the twenty sparrow classes of CUB-200. On the left, the classifier prototypes $\boldsymbol{c}_j$ (crosses) are concentrated near the center, despite their involvement in the UMAP training. To address differences in norm, $\boldsymbol{c}_j$ scaled to the norm of their respective class are also plotted as triangles. These points remain dispersed and are seldom close to their corresponding class. In contrast, in the right plot, the KMEx prototypes (squares) are located within their respective clusters. It remains uncertain whether their central concentration within classes is an artifact of the UMAP projection or reflects their distribution in the original 512-dimensional embedding.

4 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 Features Activations Let us first decompose the encoder into B blocks of layers:

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

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'.

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

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

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

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

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

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

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

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 $\check{\boldsymbol{s}}$ 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 $\check{\boldsymbol{h}}^{(b:)}$ and $\check{\boldsymbol{s}}^{(b:)}$ 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 $\boldsymbol{c}_j$.

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

The distinction lies in the normalization by the squared norm, which ensures that $\text{att}(\boldsymbol{c}_j^T, \boldsymbol{c}_j) = 1$ 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, $\check{\boldsymbol{s}}^{(b:)} \in \mathbb{R}^{W_b \times H_b \times K}$.
2. Upsample $\text{att}^{(b+1:)} \in \mathbb{R}^{W_{b+1} \times H_{b+1}}$ to $\text{att}^{(b+1:,\text{up})} \in \mathbb{R}^{W_b \times H_b}$.
3. Compute $\text{att}^{(b:)}$ as the segment-wise average of $\text{att}^{(b+1:,\text{up})}$ based on $\check{\boldsymbol{s}}^{(b:)}$.

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

Accuracy In the following, the 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.

Although B4 utilizes information close to the classifier, its prototypical classifier relies on cluster distributions rather than vector operations. Therefore, it is remarkable that its performance matches that of the backbone network (Table 2). As for B234, the inclusion of information at multiple depths renders the classification signal less distinct, leading to reduced performance.

Figure 2: An illustration of the interpretation process for the proposed methods B4 (left) and B234 (right) is provided. From left to right and top to bottom: input image of a red-bellied woodpecker from the CUB-200 dataset; explanation map computed using the feature activations of block 4 (left group) and blocks 2, 3, and 4 (right group); feature attribution map where red and blue shades indicate, respectively, greater and lesser importance to the backbone's prediction. Below, the closest patches in the training set for each class prototype are shown. The border color of the patches corresponds to the segments of the explanation maps.

Interpretation Process Figure 2 illustrates the interpretation process of the proposed method. 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 the visualization of the nearest segment to associated prototype, using a consistent color scheme. The feature importance map highlights the relevance of each segment with respect to the classifier of the backbone model. Incorporating shallower feature activations enhances the resolution and detail of the map. Indeedn, explanations produced by B4 necessitate careful attention to the color scheme for accurate patch interpretation, while the original scene remains more discernible in the explanations generated by B234.

5 Conclusion

This study presents a formal framework for the post-hoc reinterpretation of standard CNNs as self-explainable models. The commutativity of the average pooling and of the linear layer allows to show that CNNs are mechanistically analogous to prototype-based SEMs. However, the column vectors of the classifier which serve as prototype do not align with the embedding undermining their representativeness and thus the trustworthiness of the operation. Learning the prototypes using k-means proves to be an efficient workaround. By combining less-compressed feature activations across different depths, our method produces detailed explanation maps supported by semantically relevant prototypes. In line with the desire not use complex and not well understood methods, we introduced a variant of CAM that approximates the importance of each segment with respect to the backbone's classifier. The results suggest that the perceived "black-box" nature of CNNs is not an inherent structural property, but rather a characteristic that can be mitigated through rigorous reinterpretation of their existing operations.

A primary limitation of the proposed method is that training prototypes using k-means clustering does not scale efficiently. Nevertheless, preliminary studies suggest that performance remains stable even when only a limited subset of the training data is used. The sustained performance of the prototypical classifier, despite its reliance on cluster distributions, calls for further investigations.

References

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

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.

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

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.

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.

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.

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.