Interpretable AI for Medical Object Detection: A Post-Hoc Approach with YOLOv8 and FAISS

Alan Baxley
The Pennsylvania State University
ajb8481@psu.edu

Truong Xuan Tran
The Pennsylvania State University
truong.tran@psu.edu

***Abstract*—This study proposes a novel hybrid approach to enhance the interpretability of convolutional neural networks (CNNs) in medical image analysis. By integrating a wrapper class with a YOLOv8 model, the proposed method leverages feature extraction and dimensionality reduction to map training set images to a vector space optimized for similarity comparisons. Utilizing Facebook AI Similarity Search (FAISS) for nearest neighbor retrieval, this method provides post-hoc interpretability through comparisons of inference image features with training samples. Using the BR35H brain tumor dataset, we identified that layer 210 of YOLOv8m, when reduced to 30 PCA dimensions, provides the most reliable feature representations for similarity comparisons. This approach demonstrates promise for improving model transparency in medical imaging while maintaining predictive performance, offering a deployable solution for existing CNNs.**

I. Introduction

Medical imaging techniques have been a foundational diagnostic tool for clinicians as they provide a non-invasive means of looking inside the patient's body to assess patient health. With the ever increasing amounts of patient data being generated with more advanced imaging techniques, the need for computer assisted diagnosis to assist clinicians has never been greater [1]. Given this demand, widespread adoption of deep learning algorithms in healthcare seems inevitable. Indeed, the anticipation for AI to make revolutionary breakthroughs in the healthcare domain is high due to the recent successes in other fields and convolutional neural networks (CNNs) in particular are best suited to meet these expectations however, their "black box" nature is proving to be a persistent obstacle that is preventing widespread use and adoption in medical image analysis [1], [2].

CNNs are a type of deep learning algorithm ideally structured for analyzing medical images. They typically consist of sequential layers that can analyze and detect patterns in images through the application of linear and non-linear transformations. These transformations project the original input image, typically represented as matrices consisting of channels of pixels, into unique vector spaces, also known as latent or feature spaces. Each layer has its own unique feature space, and the shape and dimension of these feature spaces are determined by the weights (linear transformation) and the activation function used (non-linear transformation). The weights are adjusted during training through backpropagation and gradient descent, minimizing the loss function to improve the model's accuracy [3]. This process allows CNNs to learn hierarchical features from the input data, making them particularly effective for tasks such as image classification, detection, and segmentation in medical imaging.

While the architecture and mathematics of CNNs are well understood, they are still considered to be a "black box" regarding how their output is obtained due to the sheer complexity of the high-dimensional feature spaces that CNNs generate. Activation functions introduce non-linearity into the model, adding further geometric complexity and abstraction [3]. Most models have millions of parameters, also known as weights. These weights form linear combinations with the input matrices at every layer. This, combined with the non-linear activation functions, makes traditional approaches for unboxing black boxes, such as program tracing, impractical. All of this inherent model complexity has given rise to the relatively new field of research in AI interpretability [4].

Interpretability in the domain of AI and machine learning can be defined as the science of comprehending what a model did. This comprehension must encompass the internals of a system in a way that is understandable to humans. In other words, for a system to be interpretable, it must produce descriptions that are simple enough for a person to understand using a vocabulary that is meaningful to the user [5]. Healthcare and medical imaging analysis are fields where interpretable AI is essential because clinician diagnoses of patient conditions cannot rely on black box models. This necessity is also a legal requirement in many countries; for instance, the European Union's General Data Protection Regulation (GDPR) law mandates that the decision-making process of algorithms be transparent before they can be utilized for patient care [2], [6].

Given the critical need for interpretability in medical image analysis, we propose a hybrid CNN vector library model that maps training set features to the vector space most relevant to model outputs. By measuring the similarity between inference images and training images in this feature space, we aim to provide post-hoc interpretability through the retrieval of the k-nearest training samples. This approach is attractive from a software engineering perspective due to its ease of implementation and versatility. However, assessing the extent to which this method provides useful interpretability in the domain of medical image analysis is the core question of our research. To contextualize our approach, we first review

existing interpretability methods and concepts relevant to medical image analysis. The subsequent sections will highlight the strengths and limitations of these methods, setting the stage for a detailed discussion of our implementation and case studies.

II. Related Work and Concepts

As Salahuddin et al. in their paper, "Transparency of deep neural networks for medical image analysis: A review of interpretability methods" establish, there are numerous approaches to interpretability for CNNs and to cover all of them is beyond the scope of this work. Instead, we will focus on the approaches that are most similar to our own proposed hybrid CNN vector library approach as well as the mathematical concepts utilized in our approach.

A. Attribution Maps

Attribution maps, also known as saliency maps, are a method for interpreting CNN output by highlighting the regions of an input image that most influence the model's prediction. These methods generate heat maps that visualize the importance of each pixel or region in the decision-making process. Grad-CAM (Gradient-weighted Class Activation Mapping) is a widely used technique in this category.

Grad-CAM produces visual explanations for CNN predictions by computing the gradient of the target class score with respect to the feature maps of a convolutional layer. These gradients are then weighted by the average gradient value, producing a coarse localization map that highlights important regions in the image. This method helps in understanding which parts of the image contribute most to the prediction, making it particularly useful in medical image analysis where identifying critical areas, such as tumors, is essential [7].

Another method, SmoothGrad, enhances the interpretability of saliency maps by reducing noise. It achieves this by averaging the saliency maps generated from multiple noisy versions of the input image. This results in smoother and more reliable heatmaps, which can be crucial for interpreting medical images where precision is key [8].

Despite their usefulness, attribution maps have limitations. They often provide only a high-level view of the model's decision process without detailed insights into how specific features contribute to predictions. Additionally, the interpretability of these maps can be subjective because the highlighted regions might not always align with clinically relevant features.

B. Latent Space Interpretation

Latent space interpretation involves analyzing the high-dimensional feature spaces generated by CNNs to uncover the factors of variation that the network has learned. Techniques such as Principal Component Analysis (PCA), Canonical Correlation Analysis (CCA), and t-distributed Stochastic Neighbor Embedding (t-SNE) are commonly used to visualize these latent spaces in two dimensions.

PCA reduces the dimensionality of the feature space by projecting it onto the principal components that capture the most variance in the data. More formally, PCA obtains the k largest orthonormal eigenvectors based on the corresponding eigenvalues by performing eigenvalue decomposition on the covariance matrix of the data. These eigenvectors, also known as principal components, serve as the basis for the new vector space that the original data is projected onto [9]. The covariance matrix,obtained from the normalized data, consists rows and columns where each row is an observation and each column is a feature. The number of rows represents the dimension of the feature space and the number of columns represents the number of features.

Let $\mathbf{X}$ be the centered data matrix and n be the number of observations. Then the covariance matrix $\mathbf{C}$ can be represented as:

$$C = \frac{1}{n-1} X^T X \tag{1}$$

Performing eigenvalue decomposition on the equation:

$$Cv_i = \lambda_i v_i \tag{2}$$

provides the eigenvectors of $\mathbf{C}$ from which the matrix $\mathbf{V}$ can be constructed. Let $\mathbf{W}$ be the matrix whose columns are the k eigenvectors with the largest eigenvalues:

$$W = [v_1, v_2, \ldots, v_k] \tag{3}$$

Then the original data projected onto the reduced dimension vector space can be represented as $\mathbf{Y}$:

$$Y = XW \tag{4}$$

While PCA does not provide interpretability in and of itself directly, it helps in reducing dimensional complexity. In medical image analysis, PCA, in conjunction with other methods, can be used to visualize the relationships between different disease states or to identify patterns that are not immediately apparent in the raw data [10].

CCA identifies and measures the linear relationships between two sets of variables by finding linear combinations of the variables that optimize the correlation between the two sets. Finding this correlation involves obtaining the covariance and cross variance matrices and solving the general eigenvalue equation [11].

Let $\mathbf{X}$ be an $m \times n$ data matrix and $\mathbf{Y}$ be an $m \times p$ data matrix. The covariance matrices Σ_{XX} and Σ_{YY}, and the cross-covariance matrix Σ_{XY} are defined by Weenik in "Canonical Corelation Analysis" [12] as follows:

$$\begin{aligned} \Sigma_{XX} &= \frac{1}{m-1} X^T X \\ \Sigma_{YY} &= \frac{1}{m-1} Y^T Y \\ \Sigma_{XY} &= \frac{1}{m-1} X^T Y \end{aligned} \tag{5}$$

Solving the following generalized eigenvalue problems:

$$(\Sigma_{XY}\Sigma_{YY}^{-1}\Sigma_{XY}^{T} - \rho^2\Sigma_{XX})\mathbf{a} = 0$$
$$(\Sigma_{YX}\Sigma_{XX}^{-1}\Sigma_{YX}^{T} - \rho^2\Sigma_{YY})\mathbf{b} = 0. \tag{6}$$

Provides the eigenvectors, $\mathbf{a}$ and $\mathbf{b}$, for $\mathbf{X}$ and $\mathbf{Y}$, respectively [12]. The canonical variables can then be computed as:

$$\mathbf{u} = \mathbf{X}\mathbf{a}, \quad \mathbf{v} = \mathbf{Y}\mathbf{b} \tag{7}$$

With $\mathbf{u}$ and $\mathbf{v}$ obtained, the Pearson correlation between these canonical variables is determined, which gives us the canonical correlations ρ. These canonical correlations provide insight into the strength and nature of the relationships between the two sets of variables.

t-SNE is a powerful tool for latent space interpretation, mapping high-dimensional data to a lower-dimensional space while preserving local structure and revealing clusters of similar instances. This technique is particularly useful for understanding how CNNs group similar features and for highlighting the underlying structure of the data. By reducing high-dimensional feature vectors extracted from a CNN to two or three dimensions, t-SNE helps maintain the local relationships between data points, making it easier to identify clusters and patterns that might not be evident in higher dimensions. It is especially valuable for visualizing how similar instances group together, providing insights into the internal structure of the data and the model's decision-making process [13].

However, latent space interpretation methods are primarily exploratory and do not directly link specific features to model outputs. They provide a broader view of the feature space but may require further analysis to draw actionable insights for clinical applications.

C. Case-Based Models

Case-based models, such as Prototype Networks (ProtoPNet), provide interpretability by comparing new instances to specific examples from the training set. ProtoPNet offers an intuitive and easily understood method for clinicians by leveraging case-based reasoning to make predictions based on the similarity of input image patches to learned prototypes. ProtoPNet works by training a Convolutional Neural Network (CNN) to learn a set of prototypes for each class. These prototypes are vectors in the feature space produced by the output of the CNN backbone. During training, the prototypes are periodically projected onto training images of the same class to ensure they represent typical features of that class. The process involves adjusting the prototypes to minimize the distance between them and the patches from training images [14].

Mathematically, each prototype p_k is adjusted to minimize the distance to the patches in the feature map F of the training images:

$$p_k \leftarrow \arg\min_{i,j} \|p_k - F_{i,j}\|^2 \tag{8}$$

where $F_{i,j}$ represents the patch centered at location (i, j) in the feature map F [14]. The adjustments to the prototypes are done through stochastic gradient descent with the loss function incorporating a cluster cost that encourages the prototypes to be close to some latent patch in each training image of corresponding classes [14].

During inference, ProtoPNet compares patches of the input image to the learned prototypes. The model computes similarity scores between the input patches and the prototypes. Predictions are made based on these similarity scores, identifying which prototypes are most similar to the input image.The prototype unit g_{p_j} computes the similarity score between a patch $\tilde{z}$ in the feature map and the prototype p_j. The function g_{p_j} is defined as:

$$g_{p_j}(z) = \max_{\tilde{z} \in \text{patches}(z)} \log\left(\frac{\|\tilde{z} - p_j\|^2 + \epsilon}{\|\tilde{z} - p_j\|^2 + 1}\right) \tag{9}$$

where $\|\tilde{z} - p_j\|^2$ represents the Euclidean distance (2-norm) between the patch $\tilde{z}$ and the prototype p_j, and ϵ is a small positive constant to ensure numerical stability [14]. Therefore, ProtoPNets are able to reference their learned prototypes to provide inherent interpretability.

Case-based reasoning is particularly valuable in medical image analysis as it aligns with the clinical decision-making process. Clinicians often rely on comparisons to known cases to diagnose new patients. By presenting representative prototypes, ProtoPNet helps build trust in the model's predictions and enhances transparency, even though these prototypes are not direct examples from the training set, a crucial difference from our hybrid CNN vector library model. ProtoPNet's validity and effectiveness serve as a foundational basis for our work as it is the most closely related interpretability method to our own approach.

III. Architecture

Our hybrid approach utilizes the Facebook AI Similarity Search (FAISS) vector library along with a wrapper class that is initialized with a trained CNN and the dataset it was trained on.

A. Facebook AI Similarity Search (FAISS)

FAISS is an industrial-grade library for approximate nearest neighbor search (ANNS) of vectorized data. The basic structure of FAISS is the index which can store vectors progressively. At search time, a query vector is submitted to the index which then in turn returns the database vector that is closest to the query vector in terms of Euclidean distance [15].

FAISS employs an L2 similarity search algorithm to compute the Euclidean distance between vectors, making it particularly effective for identifying similarities in high-dimensional

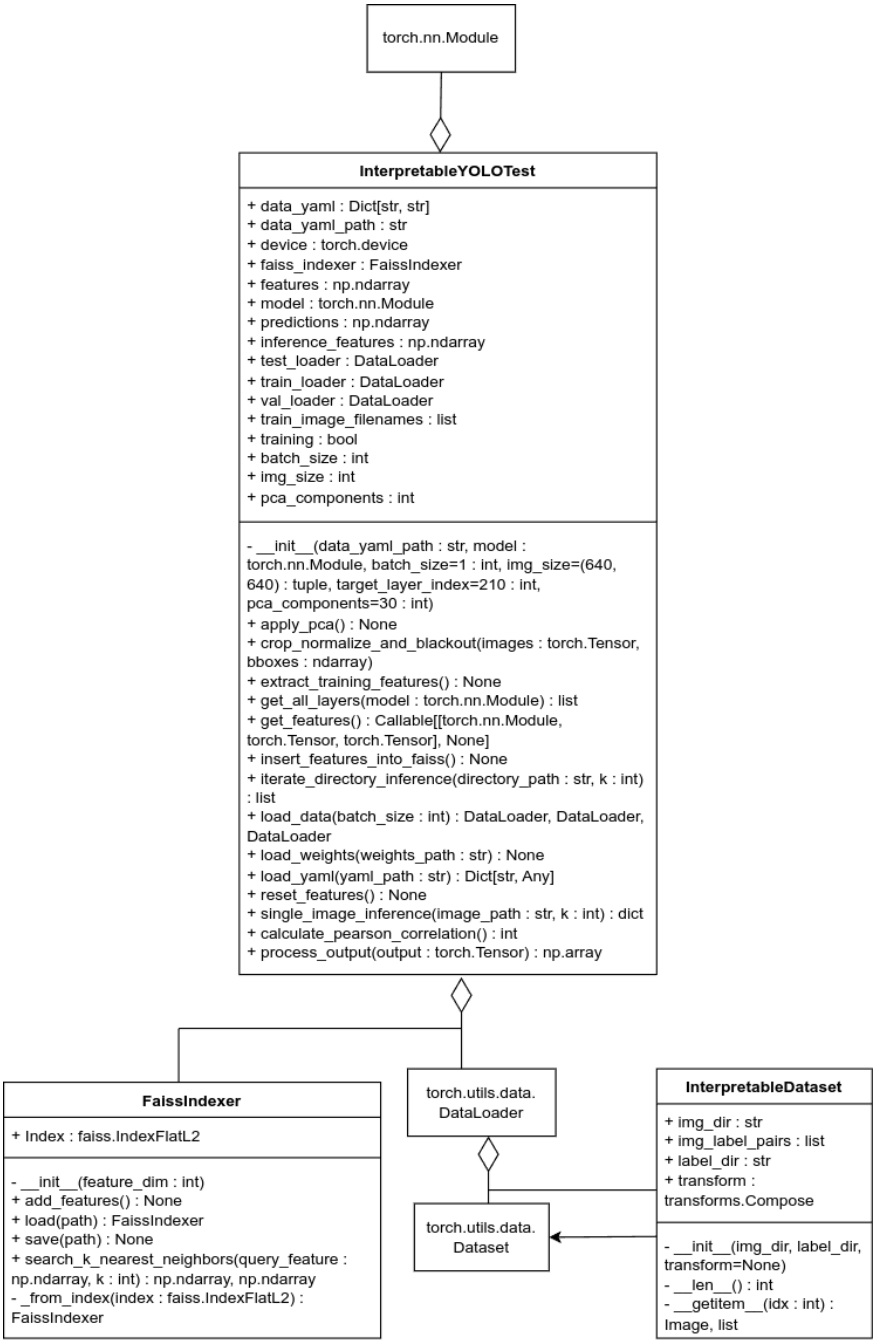

Fig. 1. Class diagram of Interpretable Wrapper classes.

feature spaces generated by CNNs. The library supports various types of indices, including flat (brute-force search), inverted file, and HNSW (Hierarchical Navigable Small World) graphs. Flat L2 indexing is the default indexing and the one we used for our hybrid approach. It calculates the top-k smallest distances to a query via a binary heap on the CPU typically, making it widely accessible to device's lacking a meaningful GPU [15].

FAISS's ability to perform rapid similarity searches enhances the practicality of using large training sets for model interpretability. Integrating FAISS in our hybrid CNN vector library model allows for efficient mapping of training set features to a given CNN layer's vector space, enabling post-hoc interpretability through the retrieval of the k-nearest training samples to an inference image.

B. CNN Wrapper Class

As illustrated in Figure 1, our wrapper class is designed to enhance the interpretability of a trained YOLOV8 model by encapsulating the model within a structured interface.

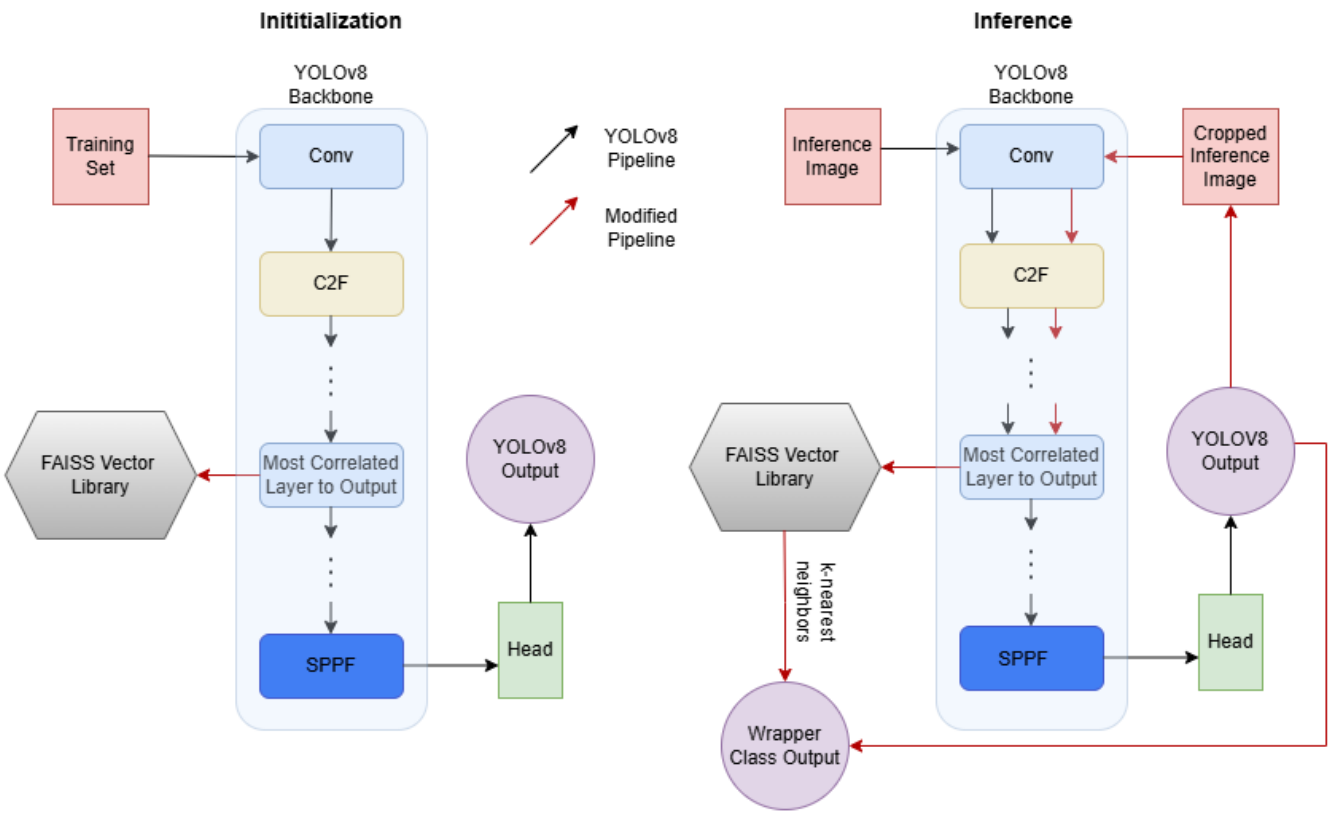

Fig. 2. Pipeline of Interpretable Wrapper class using YOLOv8.

The class constructor accepts several parameters: a YOLOv8 model, a string specifying the path to the training dataset, an integer indicating the index of the layer from which to extract features, and an integer representing the number of dimensions to reduce to using PCA. Within the class, there are methods to:

- Obtain a list of all model layer indices.
- Extract training features from the specified target layer.
- Apply PCA to the extracted features for dimensionality reduction.
- Store the features in a FAISS vector library for efficient similarity search.
- Perform inference on new images.
- Calculate the Pearson correlation between bounding box confidence scores and the target layer's feature dimensions.
- Crop images based on bounding box coordinates, enabling more relevant similarity comparisons in vector space.

As shown in Figure 2, the wrapper class, upon instantiation with a trained YOLOv8 model and its dataset, extracts training set features from a predetermined target layer and stores the vectorized training images in a FAISS index. Features are extracted from the target layer using PyTorch hooks, which are set during the class's initialization. The hook is triggered at the target layer every time an image is forwarded through the CNN, capturing the features produced by that layer. To ensure that only relevant regions contribute to the vector space, the YOLOv8 pipeline is modified so that the area of interest inside the ground truth bounding box coordinates is extracted, while all regions outside the bounding box are blacked out. This removes location bias while preserving tumor size as a factor. The normalized bounding box crops are then forwarded through the CNN, with the PyTorch hook capturing the vectorized features of each cropped training image. These modifications are applied consistently to inference images, where prediction bounding box coordinates are used for cropping, ensuring alignment between training and inference processes. By focusing the similarity comparisons on task-relevant regions, as determined by YOLOv8 model, this approach enhances the interpretability of the vector space.

During inference, the inference image is forwarded through the CNN, and, similar to the training set, the PyTorch hook captures the vectorized feature of the inference image. The FAISS library is queried with the inference image features to retrieve the k-nearest training samples. The model's normal prediction is returned along with the image IDs of the k-nearest training samples and their distances from the inference image.

IV. Materials and Methods

We instantiated one demonstration model with YOLOv8m for object detection. We trained the YOLOv8m model on the BR35H brain tumor dataset [16]. This dataset was chosen

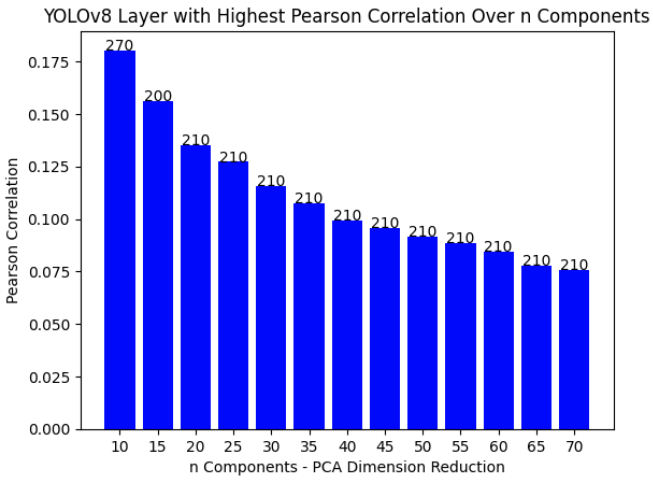

Fig. 3. YOLOv8: Most Correlated Layer by Dimension

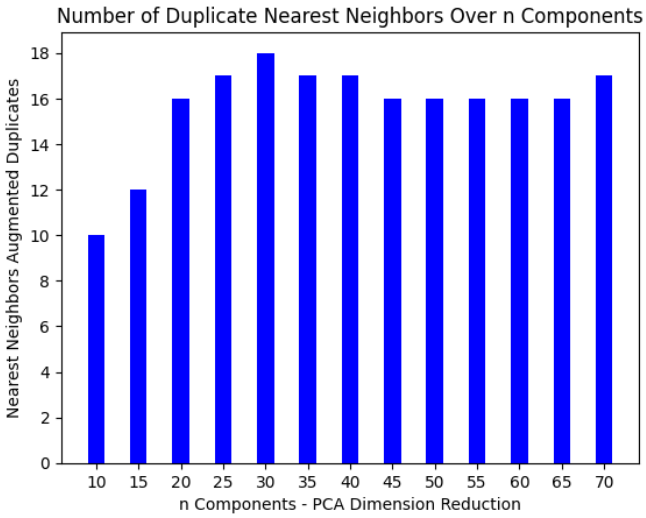

Fig. 4. YOLOv8: Number of Correctly Matched Augmented Duplicates

for it's single class simplicity allowing for interpretability performance to be the focus of the experiment. The YOLOv8m model achieved a mAP50 of 0.942 and a mAP50:95 of 0.741 with the default parameters on the BR35H dataset.

We conducted an experiment with our wrapper class for the YOLOv8m model to determine the optimal number PCA components to reduce to as well as the most relevant layer to extract image features from. Hardware for all tests consisted of the following: RTX 4070 GPU with 12 GB VRAM, 32 GB DDR4 RAM, i7-10700k CPU, and Linux Ubuntu 24.04 operating system.

To determine the optimal feature dimension and model layer for extraction, we employed two heuristics. The first heuristic involved calculating the Pearson correlation between the mean confidence score for each sample and the corresponding extracted feature vectors from the target layers of the model. This correlation assesses the linear relationship between the model's feature representations at a specific layer and its prediction confidence, indicating how well the features align with the model's decision-making process. The second heuristic involved identifying augmented duplicate images in the test set and verifying that the nearest neighbors retrieved from the training set corresponded to these duplicates.

For the first heuristic, we applied PCA to reduce the feature dimensionality, systematically evaluating every 5th dimension from 10 to 70, resulting in 13 dimensions. For each dimension, we extracted features from every 10th layer starting from layer 140, as well as the last three layers before the fully connected layer (layers 292, 293, 294), totaling 19 layers. To calculate the Pearson correlation for a given layer, we compare the mean confidence scores across all bounding boxes for each image with the feature values of each dimension in the layer's output for the same images. The process iterates over all feature dimensions of the layer, computing the Pearson correlation coefficient between the feature values and the confidence scores. These correlation coefficients are then averaged (using their absolute values) to provide an overall measure of how strongly the layer's features correlate with the model's output confidence. This comprehensive evaluation allowed us to identify the layer that exhibited the highest Pearson correlation with the mean confidence scores for each tested dimension.

The second heuristic was tested by using the most correlated layer identified for each dimension. We identified three images from the test set (images A, B, and C) that had augmented duplicates in the training set (each with six duplicates). For each combination of dimension and its corresponding most correlated layer, we extracted features and retrieved the six nearest neighbors for images A, B, and C. We then evaluated whether these nearest neighbors corresponded to the known duplicate images, thereby validating the effectiveness of the selected feature dimensions and layers.

V. Results

This section presents the outcomes of our experiment which highlights the correlation between model layers, dimension, and outputs for YOLOv8m.

As shown in figure 3, layer 210 consistently had the highest Pearson correlation coefficient across all dimensions tested with the exception of dimensions 10 and 15 which had layers 270 and 200 as being the most correlated, respectively.

Figure 4 shows the results of testing each of these dimensions with their corresponding most correlated layer against the augmented duplicate sets of images A, B, and C. Only dimension reduction to 30 with layer 210 was able to correctly match all 18 augmented duplicate images. That is, the nearest 6 neighbors when running inference on images A, B, and C were the 6 duplicate images for each image.

Using both heuristics, we conclude using layer 210 with PCA dimension reduction to 30 dimensions provides the most reliable similarity comparison in vector space for the YOLOv8m model.

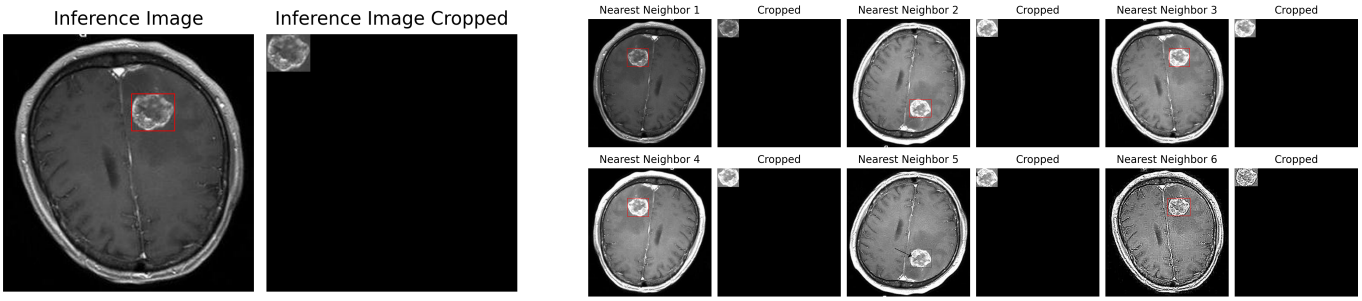

Fig. 5. Results for Image A - Layer 210, 30 PCA components: (Left) Inference Image (Right) Nearest Neighbors

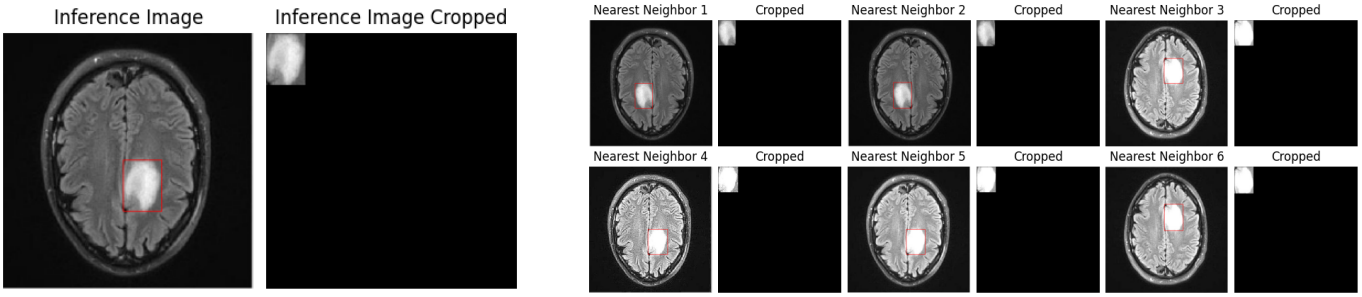

Fig. 6. Results for Image B - Layer 210, 30 PCA components: (Left) Inference Image (Right) Nearest Neighbors

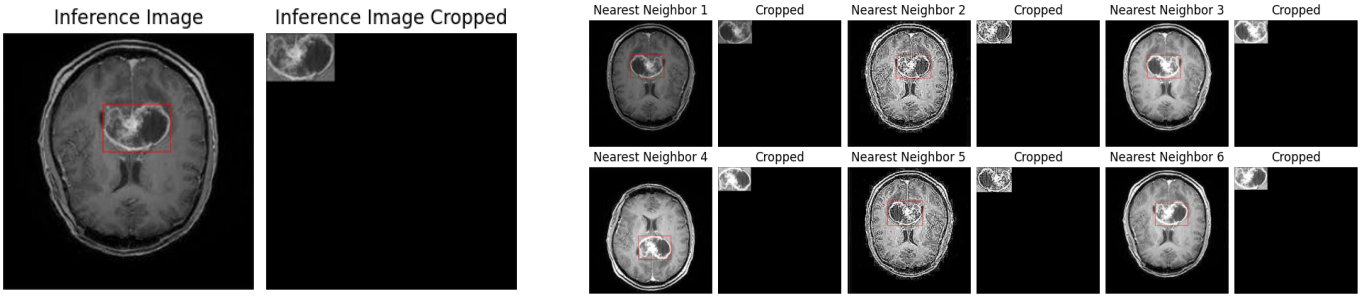

Fig. 7. Results for Image C - Layer 210, 30 PCA components: (Left) Inference Image (Right) Nearest Neighbors

VI. Discussion

The results from our tests indicate that layer 210 of the YOLOv8m model was consistently the most correlated layer across various dimensions tested, with the exception of dimensions 10 and 15. This finding is unexpected, given that layer 210 is a 2D convolutional layer with both input and output channels set to 192, and is relatively far removed from the fully connected layers typically associated with higher-level feature representations. Layer 210's role within the YOLOv8m architecture suggests that it is part of the model's backbone, which is responsible for extracting intermediate features that are essential for object detection but not necessarily final decision-making.

The high correlation observed at this layer may imply that the features it captures are particularly relevant for the model's confidence in its predictions. These features might represent an optimal balance between spatial and semantic information, which are critical for the model's decision-making process. The fact that this layer is part of the backbone further supports the idea that the backbone's mid-level features are key contributors to the model's overall performance.

Moreover, when testing different PCA dimensions, only the 30-component reduction at layer 210 was able to correctly match all 18 augmented duplicate images from the test set. This reinforces the notion that the features extracted at layer 210, when reduced to 30 dimensions, provide a reliable representation in the vector space for similarity comparisons. It is likely that these features capture essential characteristics of the images that align closely with the ground truth, making them highly effective for nearest neighbor retrieval.

The unexpected yet consistent performance of layer 210 underscores its importance within the YOLOv8m model for capturing features that are both robust and relevant to the model's confidence. Despite the fact that vector space similarity does not always translate to similarity in the original data space, this layer's ability to maintain high correlations and

accurately match augmented duplicates suggests that it plays a pivotal role in the model's interpretability when used with the proposed CNN vector library approach. Further investigation into why this layer performs so well could provide deeper insights into the relationship between intermediate convolutional layers and model interpretability in object detection tasks.

Our findings suggest that the hybrid CNN vector library approach holds promise for improving the interpretability of CNN models in medical image analysis with added benefit of being deployable to already trained CNNs due to the Post-Hoc nature of our method.

VII. Conclusion

In summary, this study introduces a hybrid CNN vector library model that demonstrates promising potential for enhancing the interpretability of CNNs in medical image analysis. Our findings suggest that by mapping training images to a vector space most relevant to the model's output, it is possible to achieve post-hoc interpretability while maintaining the model's predictive performance. Our experiment identified layer 210 of YOLOv8m, a backbone layer, as particularly effective in providing reliable similarity comparisons when coupled with PCA reduction to 30 dimensions. Future work should focus on extending this approach to datasets with multiple classes to evaluate its robustness and generalization ability in more complex image analysis tasks. Such advancements could potentially improve the interpretability of CNNs in critical domains like medical image analysis, where transparency and trust are paramount.

Supplementary Material and Code: The supplementary material and code are available at https://github.com/ajbax-cmd/Interpretable-Brain-Tumor-Detection

Acknowledgments

This work was funded by the Pennsylvania State University Multi-Campus Research Experience for Undergraduates.

References

[1] H.-P. Chan, R. K. Samala, L. M. Hadjiiski, and C. Zhou, "Deep learning in medical image analysis," in *Deep Learning in Medical Image Analysis*, G. Lee and H. Fujita, Eds. Springer International Publishing, vol. 1213, pp. 3–21, series Title: Advances in Experimental Medicine and Biology.

[2] Z. Salahuddin, H. C. Woodruff, A. Chatterjee, and P. Lambin, "Transparency of deep neural networks for medical image analysis: A review of interpretability methods," vol. 140, p. 105111. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0010482521009057

[3] M. Lee, "The geometry of feature space in deep learning models: A holistic perspective and comprehensive review," vol. 11, no. 10, p. 2375. [Online]. Available: https://www.mdpi.com/2227-7390/11/10/2375

[4] W. Samek, T. Wiegand, and K.-R. Müller, "Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models." [Online]. Available: http://arxiv.org/abs/1708.08296

[5] L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, and L. Kagal, "Explaining explanations: An overview of interpretability of machine learning," in *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*. IEEE, pp. 80–89. [Online]. Available: https://ieeexplore.ieee.org/document/8631448/

[6] R. Hamon, H. Junklewitz, I. Sanchez, G. Malgieri, and P. De Hert, "Bridging the gap between AI and explainability in the GDPR: Towards trustworthiness-by-design in automated decision-making," vol. 17, no. 1, pp. 72–85. [Online]. Available: https://ieeexplore.ieee.org/document/9679770/

[7] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual explanations from deep networks via gradient-based localization," vol. 128, no. 2, pp. 336–359. [Online]. Available: http://arxiv.org/abs/1610.02391

[8] D. Smilkov, N. Thorat, B. Kim, F. Viégas, and M. Wattenberg, "SmoothGrad: removing noise by adding noise." [Online]. Available: http://arxiv.org/abs/1706.03825

[9] J. Shlens, "A tutorial on principal component analysis." [Online]. Available: http://arxiv.org/abs/1404.1100

[10] C. N. Neves, D. S. Da Encarnação, Y. C. Souza, A. O. R. Da Silva, F. B. S. Oliveira, and P. E. Ambrósio, "Principal component analysis in digital image processing for automated glaucoma diagnosis," in *XXVII Brazilian Congress on Biomedical Engineering*, T. F. Bastos-Filho, E. M. De Oliveira Caldeira, and A. Frizera-Neto, Eds. Springer International Publishing, vol. 83, pp. 1527–1532, series Title: IFMBE Proceedings. [Online]. Available: https://link.springer.com/10.1007/978-3-030-70601-2_224

[11] J. Rupnik and J. Shawe-Taylor, "Multi-view canonical correlation analysis," in *Proceedings of the 13th Multiconference on Information Society, IS*, Ljubljana, Slovenia, 2010, pp. 201–204.

[12] D. Weenink, "Canonical correlation analysis," *Institute of Phonetic Sciences, University of Amsterdam, Proceedings 25*, pp. 81–99, 2003.

[13] L. van der Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, Nov. 2008.

[14] C. Chen, O. Li, D. Tao, A. Barnett, C. Rudin, and J. K. Su, "This looks like that: Deep learning for interpretable image recognition."

[15] M. Douze, A. Guzhva, C. Deng, J. Johnson, G. Szilvasy, P.-E. Mazaré, M. Lomeli, L. Hosseini, and H. Jégou, "The faiss library." [Online]. Available: http://arxiv.org/abs/2401.08281

[16] A. Hamada, "Br35h: Brain tumor detection," 2020. [Online]. Available: https://www.kaggle.com/datasets/ahmedhamada0/brain-tumor-detection