ChannelExplorer: Exploring Class Separability Through Activation Channel Visualization

Md Rahat-uz- Zaman, Bei Wang, and Paul Rosen

Abstract—**Deep neural networks (DNNs) achieve state-of-the-art performance in many vision tasks, yet understanding their internal behavior remains challenging, particularly how different layers and activation channels contribute to class separability. We introduce *ChannelExplorer*, an interactive visual analytics tool for analyzing image-based outputs across model layers, emphasizing data-driven insights over architecture analysis for exploring class separability. *ChannelExplorer* begins with a dataset-level overview and progressively drills down to individual examples, summarizing activations across layers along the way. It presents these results primarily through three coordinated views: a Scatterplot View to reveal inter and intra-class confusion, a Jaccard Similarity View to quantify activation overlap, and a Heatmap View to inspect activation channel patterns. Our technique supports diverse model architectures, including CNNs, GANs, ResNet, and Stable Diffusion models. We demonstrate the capabilities of *ChannelExplorer* through four use-case scenarios: (1) generating class hierarchy in ImageNet, (2) finding mislabeled images, (3) identifying activation channel contributions, and (4) locating latent states' position in the Stable Diffusion model. Finally, we evaluate the tool with expert users.**

Index Terms—**Deep Neural Networks, neuron activations, explainable AI, interactive visualization**

I. Introduction

Deep neural networks (DNNs) achieved remarkable success across domains such as image classification, object detection, semantic segmentation, facial recognition, and generative modeling [1]. Yet, understanding how these models make decisions, beyond visualizing traditional parameters and metrics such as accuracy, loss, or ROC curves [2]–[5], remains an open challenge. Advanced visual analytics tools can help interpret what the model perceives from the input data, which remains an active field of research [2], [6].

A central challenge in this space is class separability: the degree to which internal representations of different classes are distinct. The concept of class separability is not limited to classification tasks. More broadly, it reflects a model's ability to distinguish and group input concepts in its latent space, whether as discrete categories for classification or as continuous separations that guide non-classification tasks such as image generation or super-resolution. Despite high accuracy, models often suffer from poor separability that leads to weak decision boundaries, class ambiguity, and inconsistent predictions, especially under data imbalance or noise [7]–[9]. Hence, understanding where a prediction lies within the overall distribution is essential, particularly in high-stakes domains such as safety, healthcare, and finance.

Improving separability has broad benefits. It (1) enhances *interpretability* by revealing whether the network learns meaningful, class-distinctive features; (2) produces *clearer decision boundaries* and pinpoints hard-to-distinguish classes for layers; (3) provides *indirect evidence relevant to generalization and robustness* by highlighting highly overlapping or brittle internal representations; (4) enriches *data quality* by exposing mislabeled or imbalanced samples [9]; and (5) guides *possible model refinement target identification* by informing where to retrain, regularize, or penalize ambiguous regions in the feature space [10], [11]. Finally, identifying the exact *layers* or *activation channels* where separability fails enables focused interventions such as block-specific retraining, pruning redundant filters, or targeted data curation, rather than costly model-wide retraining [11], [12].

Existing works have analyzed model internals through dimensionality reduction on activations [13], [14] and feature dissection methods [15]–[18], but these approaches are often applied in isolation, limited to specific architectures (mostly CNNs), and lack integration into a reusable workflow. More recent methods, such as attribution summarizations [19] and attribution graphs [20], [21], offer deeper insight into model mechanisms, but they still do not localize separability breakdowns to specific layers or channels, nor do they provide interactive, end-to-end support for model debugging across architectures, datasets, layers, and channels. Identifying the layer or channel responsible for separability failures enables targeted interventions, e.g., block-specific retraining or regularization, pruning redundant filters, and focused data curation, rather than expensive model-wide changes [11]. The lack of class separability research and tools motivates the need for a unified, model-agnostic tool that can both visualize class separability at multiple levels of granularity of the model.

In this paper, we examine neural network activations to improve the interpretability of class separability in DNNs with image-based layers (e.g., convolution, deconvolution, affine transformation, upsampling, and residual blocks). Beyond being specific to CNNs, we focus on a more generalized approach that supports a wider range of architectures with image-like tensors, e.g., U-Net, Generative Adversarial Networks (GANs), Stable Diffusion models (SD), and Fully Convolutional Networks (FCNs). We propose the implementation of the visual analytics tool, *ChannelExplorer*, that enables tracing separability across dataset, layer, and channel levels, facilitating all five forms of model improvement discussed above.

Our contributions are threefold.

Md Rahat-uz- Zaman is with University of Utah. E-mail: rahatzamancse@gmail.com
Bei Wang is with University of Utah. E-mail: beiwang@sci.utah.edu
Paul Rosen is with University of Utah. E-mail: paul.rosen@utah.edu

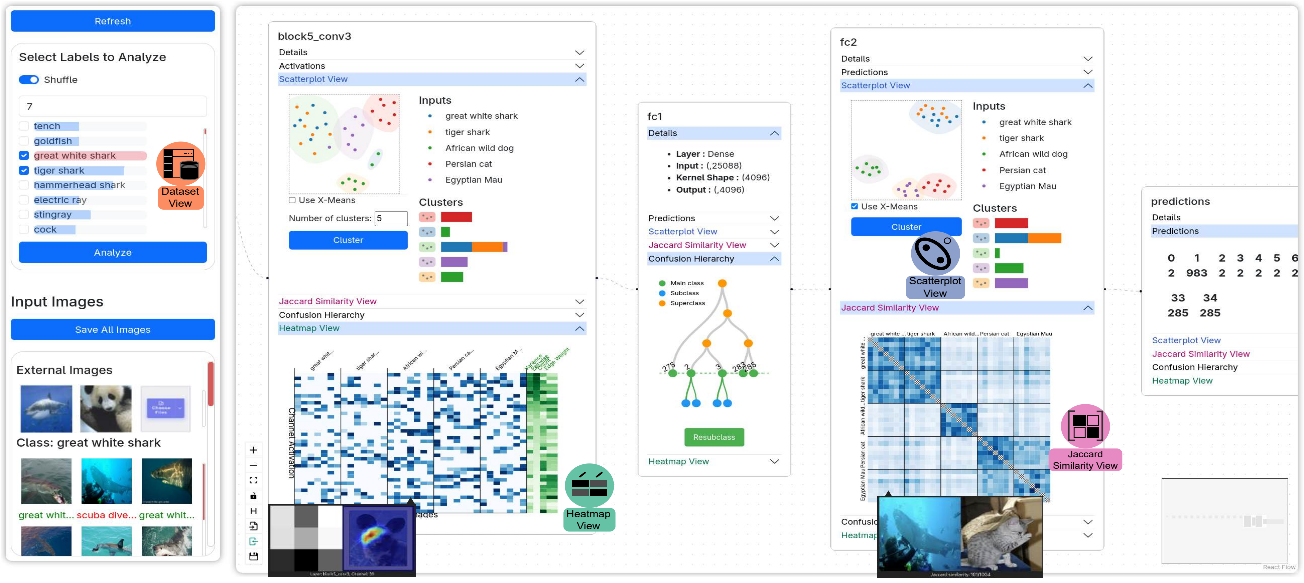

Fig. 1. *ChannelExplorer* has seven representative images loaded from five classes using Dataset View. Each layer is a node in the model's node-link diagram view. The tool has three key views: (i) Scatterplot View: 2D embeddings of activation summaries highlight class confusion patterns, e.g., overlapping clusters for shark classes (inter-class confusion) and multiple clusters within the dog class (intra-class confusion); (ii) Jaccard Similarity View: a similarity matrix showing activation overlap between inputs, e.g., high similarity between shark classes and cat classes; (iii) Heatmap View: a heatmap of activation channel summaries, enabling identification of channels that respond exclusively to certain classes and may encode class-distinctive features, e.g., activation channel for dogs' eyes. The confusion hierarchy view visualizes a class hierarchy from the layer's class confusion.

- **Enhanced interpretability.** *ChannelExplorer* highlights class separability within the activation space at each layer, making the model's decision-making process more transparent and *interpretable*. The combination of Scatterplot View and Jaccard Similarity View reveals both inter-class confusion (when similar neuron activations between classes cause confusion) and intra-class confusion (when variation in activations within a class complicates classification). It also helps machine learning (ML) experts derive a class hierarchy that pinpoints weaknesses in the model surrounding poor class separability.
- **Guided model analytics.** *ChannelExplorer* facilitates guided drill-down exploration from whole model to individual activation channels. It auto-selects representative inputs that cover the entire dataset, enables exploration at the layer level with the Scatterplot and Jaccard Similarity views, and finally provides granular details at the activation channel level with the Activation Heatmap view. By employing an overview-first, details-on-demand approach, it visualizes the contributions of layers and activation channels, highlighting their ability to separate classes and detect targeted features in the input.
- **Enriched data quality.** *ChannelExplorer* helps identify mislabeling and annotation errors. Poorly separated classes identified via the Scatterplot View and the Heatmap View often indicate mislabeling or inconsistent annotations. *ChannelExplorer* helps detect mislabeled samples for correction, leading to better *data quality*.

Finally, *ChannelExplorer* is a modern web-based and open-source tool, allowing ML experts to analyze class separability of any model with image-based layers using our provided programming API and documentation.

II. An Introduction to Neural Network Activations

A Deep Neural Network (DNN) consists of multiple layers of neurons as processing units: the *input layer* receives raw data (e.g., an image), the *hidden layers* analyze the data by detecting patterns, and the *output layer* generates the final result (e.g., a classification label). However, the interpretability of a neural network lies in its hidden layers, where the network discovers a new, hidden representation of the input at each layer.

The learned representation of images at each layer is a tensor and can be visualized as a 3D cube, where each cell represents an activation, that is, the extent to which a neuron fires. The x- and y-axes represent the spatial positions of pixels in the image, while the z-axis denotes the channels. Slicing such a cube in x- and y-axis produces vectors called *spatial activations*, and slicing through z-axis produces matrices called *channel activations* [20]; as illustrated in Fig. 2 (A and B).

Studies show that the channel activations convey information about the features the layer is attempting to detect [13], [15], [19], [22]–[24]. Earlier layers typically respond to basic features (e.g., edges, gradients, intensity variations, etc.), whereas latter layers extract more complex features that support the goal task [25]. Fig. 2 shows that channels in early layer i activate sparsely for basic textures, whereas channels in later layer j detect complex features in focused regions.

Our tool *ChannelExplorer* focuses on exploring the channel activations for enhancing the interpretability of DNNs that contain image-based layers.

III. Related Works

We discuss existing works related to class confusion analysis, class hierarchy generation, and DNN visualization tools.

Visualizing model architecture, metrics and parameters.

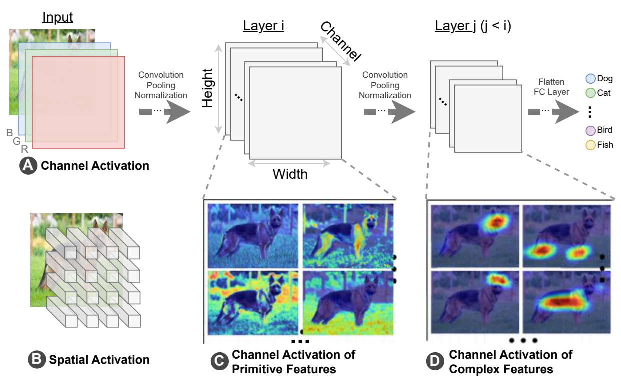

Fig. 2. (A) and (B) illustrate two approaches to analyzing image activations. We adopt the approach in (A), where each channel encodes the presence of specific features in the input. Channel activations from two different layers are shown in (C) and (D). Layer i captures low-level features (e.g., grass texture, fur texture, edges), whereas layer j captures higher-level semantic features (e.g., a dog's head or body).

Many deep learning visualization tools focus on model parameters and evaluation metrics such as precision, recall, F1-score and confusion matrix. Several tools visualize performance metrics and confusion matrices, such as ConfusionFlow [26], InstanceFlow [27], CNNComparator [28], EnsembleMatrix [29], and Neo [30], which models the confusion matrix as a probabilistic distribution with hierarchical labels. Other systems provide similar visualizations of model weights and confusion matrices at class or instance level [6], [31]–[37].

However, relying only on scores and confusion matrices can not identify generalizability for all classes. For instance, models can have accurate predictions with low confidence or overlapping activation patterns with other classes, which cannot be captured with accuracies or confusion matrices. Finally, these tools largely overlook the internal activation dynamics across layers that are critical for model transparency and interpretability.

Visualization of Raw Activation Channels. Some visualization tools directly normalize and show the activation channels. DeepVis [24], Tensorflow Playground [38] and Quiver [39] show the channels one layer at a time. However, they lack the visualizations of analytics on the activation channels.

Activation Channels as Features of a Model. Several studies propose that activation channels can be used as interpretable features in another machine-learning model to determine feature relevance [22]. Network Dissection quantified alignment between channels and semantic concepts via pixel-wise segmentation [15], while Net2Vec computed the *Intersection over Union (IoU)* score between two classes (e.g., dog and airplane) to interpret the feature similarity [16]. TCAV applied binary classification to the activations to identify the patterns (e.g., zigzag pattern) that are relevant to a specific class (e.g., zebra) [17], later extended by ACE, which clustered latent segments and applied TCAV-based importance scores [40]. ConceptExplainer used both TCAV and ACE to create an open-source tool for non-expert users [41]. Finally, ActiVis presented a tabular interface of average activations per class [42]. These methods provide useful insights, but they lack visualization at the granularity of channels or input samples, which is important to identify channels that do not help in the target task.

Dimension Reduction on Activation Channels. Some techniques reduce the dimension of activations and project them into 2D or 3D space. Activation Atlas [13] and the t-SNE visualization of CNNs [14] collected random spatial activations and visualized them in a 2D plane using UMAP and t-SNE, respectively. Each image in the 2D embedding represents the average activations of the model. Pezzotti et al. used HSNE [43] embeddings to analyze and show stable layers during training [44]. Since most of these studies focus on average class activations, the embeddings cannot visualize individual images or layers for debugging the model.

Layer Activation Maximization. Some DNN interpretation technique involves maximizing the activations of a layer by iteratively updating an input noise to turn into a feature interpretable image [23], [24]. Erhan et al. performed this using Stacked Denoising Autoencoders [23]. Plug & Play Generative Networks improved these approaches using a Conditional GAN that can produce higher-quality diverse images [45]. Instead of noise, Inceptionism modified an image (e.g., sky) using gradient ascent algorithms to fool a CNN model to predict it as a different class (e.g., dog, snail, fish, etc.) [46]. OpenAI Microscope [47] identified the highest activated neurons for inputs and visualized the features of those layers using DeepDream [46]. These visualizations are mostly used for educational or interpretative purposes, are limited to CNN models, and cannot directly identify errors or improvements in the model.

Aggregation, Selection, Filter, and Summarization of Activation Channels. Liu et al. created a datapath visualization method through layers to show how failures in early features (e.g., missing a panda's ear) propagate to later misclassifications (e.g., detecting a panda) [12]. Olah et al. focused on layers with the highest activations, generating feature-enhancing images [25], while Goh et al. grouped low-level activation channels by detected attributes such as color, brand, or emotion [21]. Hohman et al. analyzed the number of activated neurons for different classes and created activation pathways for visualizing feature identification [19] and adversarial attacks [48]. These visualizations have limited scope because many features identified by layers or channels are not explainable or have mental concepts. Molnar showed that 193 out of 512 channels in a layer of the VGG model are not interpretable [49].

Model Explainers for Educational Purpose. Tools such as CNN Explainer [50], Transformer Explainer [51], Diffusion Explainer [52], and GAN Lab [53] offer intuitive web-based interfaces aimed at educating beginners on the functionality of these models. However, their approaches are designed exclusively for educational purposes and lack utility for tasks such as model debugging or optimization.

A. Comparison with Existing Works

The area of DNN explainability is rich. While *ChannelExplorer* shares many qualities with prior works, it also provides some unique characteristics.

Architecture Overview. Most traditional visualizations represent the overall DNN using node-link diagrams and depict image-based layers in overviews with stacked images—typically displaying only a handful of activation channels (3–5 out of thousands) [54]–[56]. Some tools show layer details separately [19], [47], [57], [58], limiting simultaneous multilayer analysis. In contrast, *ChannelExplorer* uses a "Details Inside" design with collapsible panels embedded in each layer node, enabling scalable, side-by-side comparisons while preserving the network's mental model.

Improved Class Hierarchy Generation. Certain prior works are similar to isolated components of our visualization methodology, such as dimensionality reduction [14], [44] or confusion matrix visualization [26], [27]. These approaches show the interpretability of neural networks but fall short of enabling error detection across both data and model components at different levels of granularity (layers, channels). For instance, Blocks system [11] enabled users to construct a class hierarchy of only superclasses from a confusion matrix, and exploit the class hierarchy to force layers to classify a fixed number of classes. While this provides useful insights, our observations using *ChannelExplorer* reveal that earlier layers in modern deep networks only extract primitive features for downstream task-specific layers instead of directly doing classification. This renders tools, such as Blocks, not applicable as they assume early layers perform classification.

ChannelExplorer addresses this by analyzing class separability in terms of *user-defined classes* (user-defined groups of selected images as input), enabling applicability for both classification and non-classification tasks. Unlike existing works such as Blocks [11], our visualization generates a class hierarchy based on each layer and supports both subclassification and superclassification. The earlier class hierarchy is based on the presence of common primitive features, and the later layers' class hierarchy is based on task-driven class separability.

Toward Multi-level Contextual Analysis. All works discussed in Sect. III help explain neural networks but lack the ability to locate areas of interest in a model or dataset during visualization. We address this limitation by using a contextual hierarchical analysis process, where users can explore model behavior at multiple levels of abstraction—from overall class-level confusion across the model, to confusion within specific layers, down to the contribution of individual activation channels.

Transferable Model-Agnostic Debugging. These works are tailored to specific models and lack transferability to custom architectures or datasets. *ChannelExplorer* focuses on the *debuggability* to identify limitations in specific regions and is designed to work with any image-based model architecture.

In summary, although visualizing performance metrics, raw layer activations, and dimensionality reduction embeddings is not novel, *ChannelExplorer* advances prior approaches by introducing ordered summarization, class-driven abstraction, and channel-aware exploration. These additions enable the identification of class ambiguity, the discovery of latent class hierarchies, and the interpretation of channel-level contributions, capabilities that are largely absent in existing systems. Moreover, *ChannelExplorer* is a general-purpose, data-driven framework designed to support both interpretation and debugging of complex modern neural networks.

IV. Design Overview

Through an iterative design process led by a researcher with expertise in deep learning and visualization, and guided by feedback from two faculty experts in data visualization (one with applied deep learning experience), and finally a user study with a tool prototype involving nine students who took a graduate Deep Learning course, we identified three key visualization challenges associated with class separability.

A. Visualization Challenges

From our evaluation of prior work and discussions with various deep learning and visualization experts, we identified three key visualization challenges associated with class separability.

C1. Difficulty visualizing abundance of activation channels. Visualizing activation channels is a central focus of DNN interpretation. Although valuable, this approach does not scale well: analyzing results for more than a few images becomes tedious. Moreover, users must compare multiple channels to identify shared features and determine their contribution to the target task, a process that becomes increasingly challenging as the number of channels grows.

C2. Difficulty visualizing activation similarity. Only visualizing channels as images side-by-side is inadequate for perceiving their similarity. This is due to (i) difficulty of comparing multiple images side-by-side and (ii) ineffectiveness of human perception in comparing only the magnitudes from multiple images [59], [60].

C3. Difficulty identifying layers/channels contribution. Understanding how specific layers or activation channels contribute to the goal task is essential for debugging and optimizing neural networks. However, existing visualization techniques often lack mechanisms to quantify or highlight these contributions effectively. This challenge is further compounded when channels contribute redundantly or negatively to the task, making it difficult to isolate the most impactful components.

B. Visualization Goals

To address these design challenges, we have developed the following design objectives:

G1. Explore from abstract view to raw activations. To address C1, we introduce high-level abstractions over (i) activations at the layer level and (ii) classes, including both predefined dataset labels and user-defined categories, rather than presenting all activation channels simultaneously. This design supports coarse-to-fine exploration: users first navigate at the layer level and then focus on selected layers for detailed channel-level analysis. Class-level abstraction further enables the identification of salient classes, after which users can examine representative samples within a class to uncover potential areas for model improvement.

G2. Compare between summaries instead of images. Comparing channel magnitudes provides insight into their similarity, as these magnitudes reflect the strength of feature presence in the input. In line with G1, we therefore present abstract summaries of activation channels. Displaying these summaries side by side enables users to more readily perceive correlations and differences at the channel level, thereby addressing C2. This

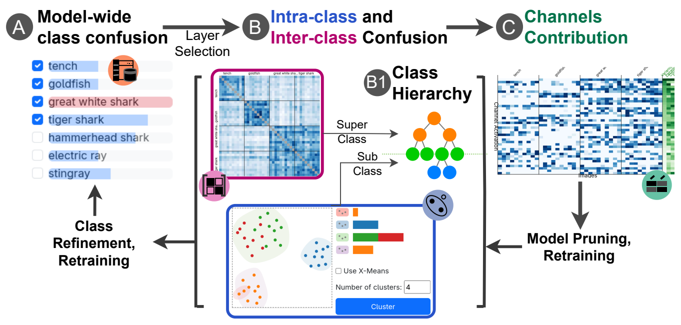

Fig. 3. Overall workflow of *ChannelExplorer*. Users start the analysis with Dataset View (A) to find classes with high confusion. Then the system shows intra-class and inter-class confusions using the two layer-level views (B), and constructs class confusion hierarchy (B1). The Heatmap View shows the activations of each channel (C).

approach is more effective than direct comparison of activation channels, as comparing aggregated values is cognitively less demanding than comparing images.

G3. Visual analytics on patterns of activations. To address C3, we aim to identify channels that contribute little to the target task by guiding users through a structured visual analytics workflow and clearly communicating the role of each visual component. Users begin by inspecting a selected layer. If its activation channels exhibit plausible behavior, they iteratively examine adjacent layers until encountering one that displays unexpected patterns, such as channels focusing on irrelevant input features. Given the hierarchical and cascading nature of DNNs, retraining the model from the identified layer to the output layer may help mitigate such undesired behavior.

System Workflow. Based on the three visualization goals, we summarize the analysis pipeline in Fig. 3.

For classification tasks, users select predefined classes of interest. For non-classification tasks, inputs can be grouped into *user-defined classes* to enable separability analysis within the tool.

Upon initialization, the system presents an overview of all classes and their similarity to a selected reference class, along with representative samples (A). The model architecture is visualized as a node-link diagram with collapsible sections per layer. The hierarchy view automatically derives a class structure based on superclass-subclass relationships inferred from class confusions (B1). Users can inspect and validate this structure by expanding summarized outputs and comparing them across views (B).

For deeper analysis, users can examine raw activations, compare input samples, and explore the latent space of individual layers (C). Selections are linked across all views, enabling consistent tracking of specific inputs throughout the analysis.

The workflow adopts an output-to-input exploration strategy: users start from the model's predictions at the output layer and progressively trace class separability backward through intermediate layers to pinpoint error sources and identify opportunities for improvement.

V. *ChannelExplorer* Visualization Design

This section discusses the design and implementation of *ChannelExplorer*. We first discuss the neural network overview and summarization in Sect. V-B and Sect. V-C. To begin, users can select classes in the Dataset View (in Fig. 1) and select a number of images that automatically represent all images of the selected classes. Then users explore the activations for those images using Scatterplot View (in Fig. 1) and Jaccard Similarity View (in Fig. 1). Then, for interesting layers, the Heatmap View (in Fig. 1) can be used to explore activation channels. The implementation details are presented in supplementary materials. A static demo with InceptionV3 model is publicly accessible in channelexplorer.netlify.app.

A. *Input and Dataset View*

As input to the model, we group samples that *should* produce similar activations or outputs. For classification tasks, these sets can simply correspond to the pre-labeled classes of the dataset. For non-classification tasks, users can define groups of inputs expected to yield comparable outcomes, which we refer to as *user-defined classes*. For example, an input image of a mountain in summer and an input of a mountain in winter produce outputs that differ in color and texture, yet both remain more similar to each other than to outputs of entirely different scenes, such as beaches or urban skylines. Grouping inputs allows users to analyze whether the model treats semantically equivalent inputs consistently in its latent space. The Dataset View can display model-wide activation similarity across all pre-labeled or user-defined classes, as well as the model's prediction for individual inputs. For clarity, the term *class* refers to both pre-labeled and user-defined classes throughout the paper.

Representative Input Selection. The input dataset can contain thousands of samples, making it infeasible to visualize all of them. Random sampling may overlook important variations or introduce bias, as different subsets can activate channels of the model in distinct patterns. To address this, we introduce a representative selection method that captures the diversity of activation patterns within each class. When a class is selected, we apply the X-Means algorithm across all layers and images to cluster activation summaries (see Sect. V-C for activation summaries). Each cluster captures a distinct activation pattern, and an exemplar input is sampled to represent it. Users may also specify K for clustering. This approach ensures that the selected representatives are not arbitrary examples, but instead reflect the diversity of activation behaviors within the class. In this way, the system becomes scalable to large datasets, and users can more efficiently explore how different subsets of images contribute to the model's internal representation of a class.

B. *Neural Network Overview*

The neural network overview shows the model as a graph, where each node represents a layer, and each edge represents connections between adjacent layers (see Fig. 4). Sugiyama layout [61] is used for the initial node placement, which can be dragged to reposition. We used a detail-on-demand approach [62] and added each view as a collapsible component in each node to achieve goal G3. To improve the visualization experience, layout navigation buttons (e.g., zoom in/out, orientation, full-screen), layout import/export buttons, minimap, and some accessibility buttons have been added.

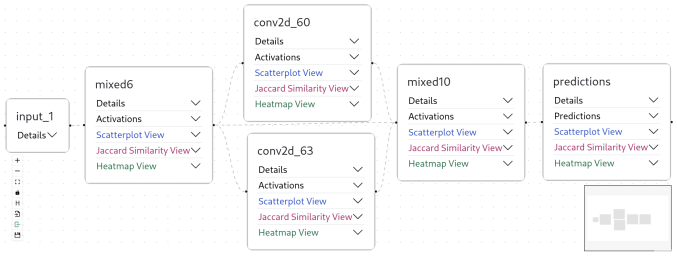

Fig. 4. InceptionV3 model visualization using *ChannelExplorer*.

C. Activation Channel Summarization

We create an abstraction for activation channels to address the visualization difficulty posed by the large number of channels in modern neural networks (**C1**). In image-based models, activation channels respond with high signed values, either positive (excitatory) or negative (inhibitory), when the corresponding feature is present in the input. For the purpose of *ChannelExplorer*, we focus on identifying *selective responsiveness* of channels to classes, rather than neuron-level causal effects on subsequent layers. Hence, we operate on absolute activation values to capture the overall response of a channel, thereby enabling a comparative analysis of spatial activations across channels within a layer.

To this end, we summarize each activation channel using magnitude-based functions that quantify how strongly a channel responds, independent of the spatial locations of high-intensity pixels. This abstraction enables meaningful comparison across channels and inputs in our visualizations.

Neural network layers often include non-linearities such as ReLU, PReLU, ELU, and SELU, as well as pooling and normalization operations, which can suppress or alter signed activations. We therefore evaluated all proposed summarization functions both before and after these non-linearities. For smaller networks (e.g., VGG16), activations are predominantly non-negative, resulting in identical summaries before and after activation functions. For larger architectures (e.g., InceptionV3), we observe that approximately 19% of activation channels contain negative values. However, across all evaluated cases, we found no qualitative differences in cluster structure, similarity patterns, or stripe formation in the three activation channel views (discussed on next subsections) when comparing pre- and post-nonlinearity summaries (see Fig. 5).

As our visual encodings operate on activation magnitudes and are designed to support comparative pattern analysis across many channels, we apply summarization functions to pre-nonlinearity activations by default. This choice preserves the full range of channel responses, including inhibitory activations that would otherwise be suppressed by rectifiers such as ReLU, while maintaining consistency across views. For completeness, we expose an API option that allows users to compute summaries after non-linearities if desired.

All summarization functions in *ChannelExplorer* operate on absolute activation magnitudes. Without this design choice, directly aggregating signed activations could introduce cancellation effects, whereby strong positive and negative responses within a channel offset one another. This may yield near-zero summary values that obscure feature selectivity and distort similarity relationships in downstream visualizations.

We acknowledge that alternative designs are possible, for example, aggregating positive and negative activations separately and visualizing them using distinct encodings. However, *ChannelExplorer* is intended to assess overall channel responsiveness in the context of class separability, rather than to disentangle excitatory and inhibitory contributions. Introducing separate summaries for positive and negative activations would double the dimensionality of the representation, complicate similarity computations, and shift the analytical emphasis toward sign-specific effects.

For our target tasks of comparing activation strength patterns across numerous channels and inputs, the magnitude-based abstraction provides a stable and interpretable measure of selective responsiveness. Users who require sign-specific analyses can readily extend the summarization API to support this functionality.

Formally, let $I \in \mathbb{R}^{w \times h}$ denote an activation channel of width w and height h, where $I(x, y)$ represents the activation value at pixel (x, y). A summarization function $S(I)$ maps I to a scalar $s \in \mathbb{R}$ representing the channel's response magnitude. Although users may define custom functions programmatically, we evaluated the following four summarization functions:

1) L2-norm: $S(I) = \sqrt{\sum_{x=1}^{w} \sum_{x=1}^{h} I(x, y)^2}$.
2) Sum of threshold: $S(I) = \sum_{x=1}^{w} \sum_{y=1}^{h} I'(x, y)$. Here, $I'(x, y) = |I(x, y)|$ if $|I(x, y)| > T$, otherwise 0. T is determined using Otsu's algorithm [63].
3) Average activation channel: $S(I) = \sum_{x=1}^{w} \sum_{y=1}^{h} |I(x, y)|/(w * h)$.
4) Maximum intensity: $S(I) = \max_{x=1}^{w} \max_{y=1}^{h} |I(x, y)|$.

We observe that summarization functions 1, 2, and 3 perform equivalently (as expected) in both the Scatterplot View and the Jaccard Similarity View (see Sect. V-D and Sect. V-E). In the Heatmap View, the sum of geometric thresholding functions also behaves as intended (see Sect. V-F), producing distinct stripe patterns across class rows. Fig. 5 illustrates the Scatterplot View and the Jaccard Similarity View, showing five activation channels after summarization.

We retain the sum of threshold as the default summarization function for consistency, although the choice is programmatically configurable. The Activation Heatmap Views generated with all summarization functions and nonlinearities are provided in Fig. 15 in the supplement.

D. Activation Distance Scatterplot View

The first step toward achieving goal **G1** is layer-level navigation. We identify layers exhibiting inter-class and intra-class confusion by visualizing distances between image activations. This analysis can also reveal potential subclass structures within the dataset, thereby helping to reduce class confusion (see Sect. VI-A for a use case).

To support this, we project image activations into two dimensions using dimensionality reduction (DR) and display the results in a 2D scatterplot (in Fig. 1). This view reveals whether a given layer effectively separates images in latent space, partially addressing goal **G3** by indicating each layer's contribution to class separability. In earlier layers, spatial proximity between points suggests that the layer captures shared low-level features.

During high-level exploration, users can expand the Scatterplot View to compare multiple layers side by side. When focusing on a specific layer, they may replace points with the corresponding input images, enabling a more detailed and comprehensive inspection of the data (see Fig. 15 in the supplement).

We employ four dimensionality reduction (DR) methods for 2D embeddings: PCA, MDS, t-SNE, and UMAP; examples are shown in Fig. 6.

The Scatterplot View emphasizes class-level variation by projecting activation distances, thereby addressing **C2**, while simultaneously supporting **G1** through an abstract, per-layer overview. In our implementation, t-SNE is used as the default DR method for three reasons. First, a survey of existing DNN visual analytics systems indicates that t-SNE is the most widely adopted technique [12]–[14], [19], [25], [44], [48]. Second, for class exploration, it often produces well-separated local neighborhoods that facilitate cluster inspection. Third, efficient implementations support interactive performance, making it practical in our setting.

We also provide PCA and MDS through the programming API as lightweight baselines. PCA is computationally inexpensive and less sensitive to initialization, which makes it useful for debugging and rapid inspection. MDS preserves pairwise distances and can better reflect global structure. Because embedding layouts can vary with initialization, we allow reseeding so that users can assess the stability of observed patterns across multiple runs.

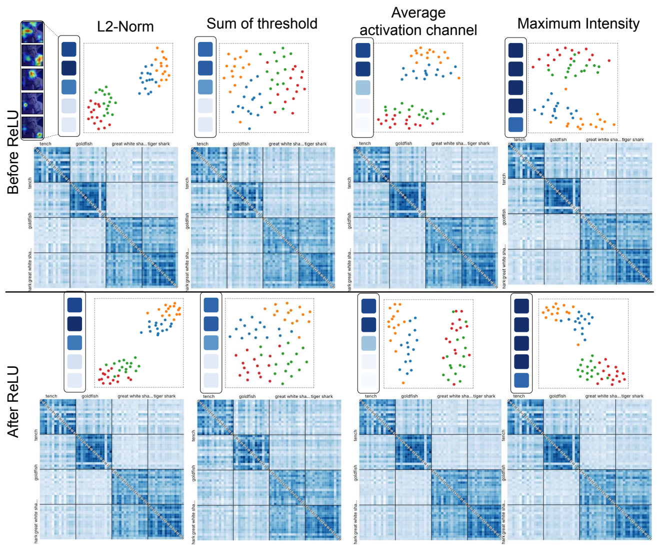

Fig. 5. Each column compares different summarization functions applied to channel activations in the Scatterplot View, Jaccard Similarity View, and activation channels. The top and bottom rows show summaries computed before and after non-linearities, consequently. All values are globally normalized for consistent color scaling.

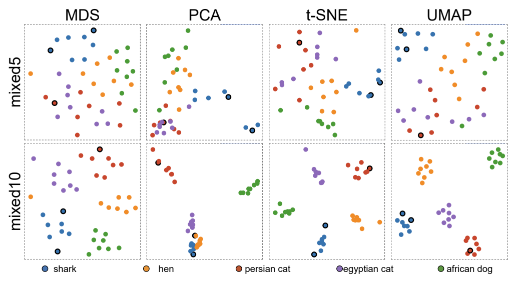

Fig. 6. Four DR methods applied to two layers' summarizations of activation channels of InceptionV3 CNN model. The last CNN layer (mixed10) shows that UMAP and t-SNE preserve the global distances between classes, whereas MDS and PCA focuse more on the local structure and variance.

We investigate two strategies for applying DR. In the first, each activation channel, represented as a $w \times h$ grid, is flattened into a vector, and all channels of an image are concatenated into a single high-dimensional representation. DR is then applied based on distances between these full activation vectors. In the second strategy, each channel is reduced to a scalar summary (e.g., $S(C^l_{ij}) \in \mathbb{R}$), forming a compact summary vector per image. DR is subsequently applied to these lower-dimensional representations. Fig. 7 compares embeddings obtained with and without channel-wise summarization. Across all DR methods, the L2-norm yields the most consistent and interpretable results; we therefore adopt it as the default distance metric for all scatterplots.

DR alone does not fully address **C2**, as projection artifacts may distort pairwise distances, causing points that are close in high-dimensional space to appear distant in 2D, and vice versa, which can lead to misleading interpretations. Although distance metrics become less discriminative in high dimensions, nearest-neighbor relationships remain meaningful when they induce stable cluster structures [64].

To mitigate structural distortions introduced by DR, we integrate X-Means clustering with Euclidean distance [65]. We also evaluated DBSCAN; however, its reliance on density estimation reduced both robustness and interpretability in high-dimensional activation spaces. When users specify a value of K, the interface instead applies k-means clustering. Although the number of clusters identified by X-Means often corresponds to the number of classes, allowing users to define K facilitates explicit assessment of class separability by testing alignment between clusters and user-provided labels.

Clusters are visualized using hull enclosures generated with Catmull–Rom curves [66]. In Fig. 7(K1), point **P** appears distant from other members of its class (purple points) in the 2D projection; however, the enclosure indicates that they remain proximal in the original high-dimensional space and form a coherent cluster. Similarly, Fig. 7(K2) shows overlapping clusters in the projection, while the X-Means enclosures reveal that they are separable in higher dimensions.

We include a horizontal bar chart to show the class distribution within each cluster. A cluster containing a heterogeneous mix of samples from many classes may indicate either substantial inter-class confusion or suboptimal centroid initialization. To differentiate between these scenarios, users can rerun the

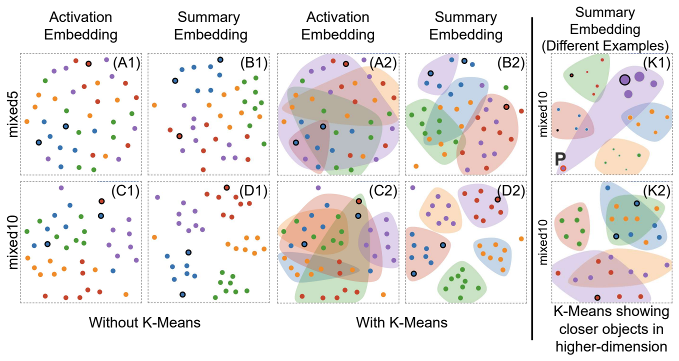

Fig. 7. (A1-D2) Comparison between Scatterplot Views using whole activation channel vs. summarization of activation channel. (K1) The purple point with a red border, **P**, is far from its cluster (purple points cluster), but k-means clustering shows that they are close. (K2) Purple and Red points' clusters are ambiguous in 2D, but k-means show they are separable in higher dimensions.

clustering with different random seeds directly through the interface.

E. Activation Jaccard Similarity View

Although DR methods and cluster enclosures provide effective abstract visualizations at the layer level (**G1**), they do not preserve the original pairwise distances between activations. Consequently, they are not reliable for precise activation comparison (**G2**). To address this limitation, we incorporate a similarity matrix that explicitly visualizes the exact pairwise distances between activations within a layer.

Similar to the Scatterplot View, we evaluate distance computation between image activations both with and without channel-wise summarization. Without summarization, we compute the pixel-wise sum of distances between corresponding activation channels for each image. This approach yields intra-class distances (diagonal entries of the similarity matrix) that are nearly indistinguishable from inter-class distances, providing limited discriminative insight. In contrast, applying summarization functions produces a clearer separation between diagonal and off-diagonal cells in the similarity matrix, enhancing interpretability. We therefore consistently employ summarization functions when constructing the similarity matrix.

We first apply the summarization functions to each channel to construct feature vectors V_{ij}^l for the ith image at layer l, as described in Sect. V-D. Next, we select the top A_η activation channels from each vector, where $A_\eta = \lceil \eta k \rceil$, $0 < \eta \leq 1$, and k denotes the total number of channels in that layer. The parameter η controls the selection granularity: larger values retain more channels, including those with weaker feature responses, whereas smaller values focus on the most strongly activated channels.

Let S_i^l denote the set of top A_η channels for the ith image at layer l. A channel is considered "fully activated" for that image if and only if it belongs to S_i^l. The Jaccard similarity coefficient matrix at layer l, denoted as J^l, is defined by

$$J_{ij}^l = \underbrace{|S_i^l \cap S_j^l|}_{\substack{\text{\# of channels} \\ \text{activated for } \textbf{both} \text{ images}}} \quad / \quad \underbrace{|S_i^l \cup S_j^l|}_{\substack{\text{\# of channels} \\ \text{activated for } \textbf{either} \text{ image}}} \qquad (1)$$

To address challenge **C2**, we visualize this Jaccard similarity matrix (in Fig. 1) as a heatmap. The color scale can be optionally normalized to the 1st–99th percentile range of the activation-channel summaries in order to suppress sporadic bright rows and columns caused by erratic input data. Rows and columns corresponding to the same class are visually grouped using separator lines. Clicking any cell in the matrix displays the two corresponding inputs along with the number of commonly activated channels (see Fig. 8).

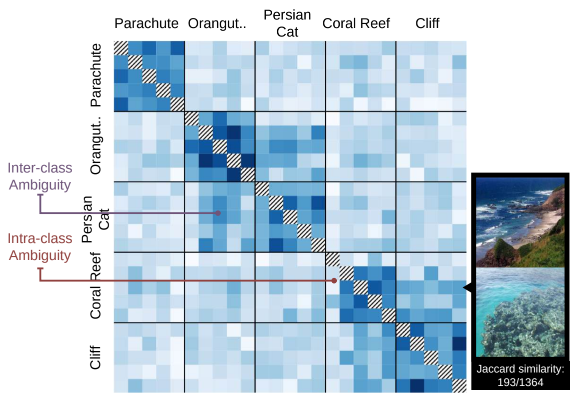

Fig. 8. Demonstration of Jaccard Similarity View. Darker colors in the distance between two classes (non-diagonal cell groups) represent inter-class confusion and lighter colors in diagonal cell groups represent intra-class confusion.

The Jaccard Similarity View supports goal **G2** by highlighting intra-class and inter-class confusion within a layer. If two classes share many fully activated channels, the corresponding off-diagonal cells appear darker, indicating high inter-class confusion. Conversely, lighter colors along the diagonal indicate greater intra-class confusion, reflecting weaker consistency of activation patterns within the same class. As shown in Fig. 8, the class *Coral Reef* exhibits relatively high intra-class confusion, and the pair *Coral Reef* and *Cliff* shows pronounced inter-class confusion at this layer.

Automatic Confusion Hierarchy Generation. Using Scatterplot View and Jaccard Similarity View, the system automatically generates a class confusion hierarchy for each layer. To construct the hierarchy, Jaccard similarity is computed between classes by averaging pairwise overlaps. Superclasses are then formed via agglomerative clustering on the similarity matrix, while subclasses are derived by further clustering activation channels within each class (as in the Scatterplot View). Without looking at the details, users can observe this hierarchy view of classes to see their activation patterns per layer (see the "Confusion Hierarchy" for layer `fc1` in Fig. 1).

F. Activation Heatmap View

After completing the first layer-level navigation, goal **G1** calls for exploration of raw activation channels. We first identify layers of interest using the Scatterplot and Jaccard Similarity views, and then examine channel-level activation patterns in the Activation Heatmap View (in Fig. 1). In this Heatmap View, each column corresponds to an input image, and each row corresponds to an activation channel.

As indicated by goal **G2**, comparing summarized activations is substantially more tractable than inspecting thousands of

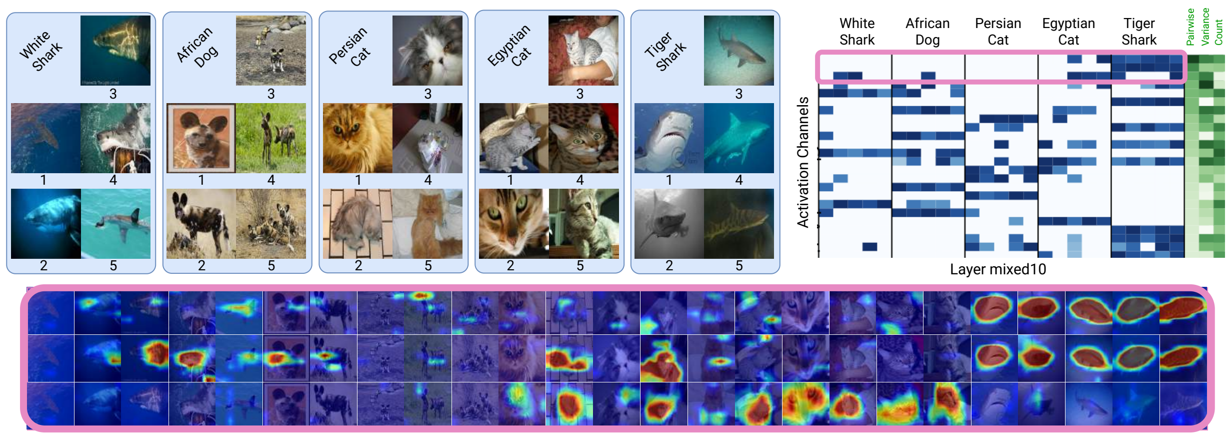

Fig. 9. Activation Heatmap View of layer mixed10 in InceptionV3. The top 3 activation channels are shown as overlays at the bottom. The heatmap's stripes (dark color in all columns of a class in a row) show that activation channels at those rows detect class-identifying features.

raw activation channels. Moreover, while the Scatterplot View addresses one aspect of goal **G3** by helping identify layer-level contributions, the Heatmap View complements this analysis by revealing channel-level contributions. It does so by enabling comparison of activation patterns within a layer across inputs from different classes.

Images (columns) are grouped by the provided classes. To reduce the dominance of low activations, activation summary values are thresholded at the 10th percentile before applying the colorscale. Clicking a cell displays the activation channel overlaid on the input image, calculated as follows.

Let the global maximum and minimum pixel values across all activation channels at layer l be defined as

$$G^l_{\max} = \max_{\forall i,j,x,y} C^l_{ij}(x,y), \quad G^l_{\min} = \min_{\forall i,j,x,y} C^l_{ij}(x,y),$$

where $C^l_{ij} \in \mathbb{R}^{w\times h}$ denotes the ith channel of the jth image at layer l.

Each activation channel is normalized to $\tilde{C}^l_{ij}(x,y)$ by scaling its values to the range $[G^l_{\min}, G^l_{\max}]$. The normalized channel is then resized to the input image resolution using bilinear interpolation. Subsequently, it is mapped to RGB space via a colormap function M_c, where $c \in \{R,G,B\}$.

Finally, the colored activation channel is blended with the corresponding input image channel values f_c to produce the output O_c, using a blending factor α. By default, we use the Jet diverging colormap and set $\alpha = 0.6$.

$$O_c(x,y) = \alpha M_c(\tilde{C}^l_{ij}(x,y)) + (1-\alpha) f_c(x,y) \qquad (2)$$

In Fig. 9, we observe the highest number of consistent stripes (dark blue for all columns in a class group) when using the sum of threshold as summarization function in multiple models. Thus, this is used as the default summarization function.

Showing thousands of rows representing activation channels can obfuscate the visualization without showing any summary or aggregated metric. Through various experiments, we add three additional metrics for ordering these activation channels: (i) variance, (ii) sum of class-pairwise distance, and (iii) edge weights from the previous layer. Let μ^l_i be the average of all $S(C^l_{ij})$, summary of ith activation channels (row i) of all n images (all j columns) at layer l. The variance is ${\sigma^l_i}^2 = \frac{1}{n}\sum_{j=1}^{n}(S(C^l_{ij}) - \mu^l_i)^2$. We assume the number of classes as C, and C^l_{ui} represent the vector of summary of ith channels of all images in class u. We compute $d(C^l_{ui}, C^l_{vi})$ to be the Euclidean distance between channel summaries of two classes. Then, the sum of class-pairwise distance is calculated by

$$\zeta^l_i = \sum_{j_1=1}^{C-1} \sum_{j_2=j_1+1}^{C} d(C^l_{ij_1}, C^l_{ij_2}). \qquad (3)$$

We calculate the L2-norms for each filter as the edge weights. These aggregated metrics are added as the last columns to the heatmaps (green columns in Fig. 9). The sum of class-pairwise distance is used as the default ordering of channels, as it shows the highest number of stripes at the last layer for models with high accuracy or good results.

For class-separating channels, each column group corresponding to a class should exhibit distinct stripe patterns. For example, in layer *mixed10* of Fig. 9, the top two rows form clear stripes for the *Tiger Shark* class, indicating that these channels respond selectively to features characteristic of *Tiger Shark*. In contrast, if an activation channel is sporadically activated across images from all selected classes, producing no discernible stripe pattern, it likely contributes little to the target task. We leverage this expected consistency of class-specific activation patterns to address goal **G3**, identifying channel-level contributions, as demonstrated in the use case in Sect. VI-C.

VI. Use Cases of *ChannelExplorer*

We discuss four use case scenarios.

A. *Scenario 1: Enhancing Class Hierarchy*

Deep neural networks (DNNs) often capture more information than required for the task [67], which can be exploited to reveal latent structure and relationships within large datasets. To demonstrate how *ChannelExplorer* supports hierarchical analysis at scale, we begin with a large multiclass setting and progressively narrow the analysis to finer-grained class relationships. Upon narrowing down to 5-10 classes, we demonstrate how the user can understand the relationships constructed automatically by utilizing the existing *ChannelExplorer* views in a CNN classifier. We also apply the visualization technique to a non-classification task, a super-resolution GAN model.

Confusion Hierarchy Discovery. Following the workflow in Fig. 3, we first select 50 semantically diverse ImageNet classes, with the help of class-wide activation similarity in the Dataset View, spanning animals, fish, and man-made objects, and load a small number of representative inputs per class using the Dataset View (step A). Rather than attempting to reason about all classes simultaneously, *ChannelExplorer* supports an overview-first, drill-down workflow in which users identify coarse structure before focusing on subsets of interest.

At this large scale, the Confusion Hierarchy View provides an immediate high-level summary of inter-class relationships at the selected layer (mixed10 of InceptionV3). Upon inspecting the groups by hovering over them to see examples in the hierarchy view, we see several coherent super-groups, including fish, birds, animals and man-made objects (demonstrated in Fig. 10). Although some have semantic similarity, these groupings are solely constructed from shared activation patterns rather than semantic labels, allowing users to reason about the model's internal organization.

Focused Analysis via Hierarchical Drill-Down. From the full hierarchy, users may choose to focus on a specific region of interest. In our example, we select a superclass containing animal categories for detailed inspection. This selection reduces the scope from 50 classes to a more manageable subset while preserving contextual relationships within the original class set. Subsequent refinements follow the same workflow. As the number of classes decreases, we recommend increasing the number of representative inputs per class to improve the stability and accuracy of the hierarchical view, since more samples provide stronger statistical support for the inferred structure.

Finally, we examine the hierarchy formed by the 5 animal classes *Ibex*, *Bighorn*, *Bison*, *Ox*, and *Cat* (top right of Fig. 11) to better understand the underlying grouping rationale. We select *Cat* as a baseline class to stabilize similarity comparisons in the Jaccard Similarity View (step B). To analyze how this confusion hierarchy is constructed, we further investigate the remaining three views.

Super-classification in ImageNet. The Jaccard Similarity View in Fig. 11 shows a high similarity between *Bison* & *Ox* and *Ibex* & *Bighorn* among 10 images per class. Hence, we merged them into two super-classes, naming *Bovid* and *Caprines*, respectively.

Subclassification in ImageNet. Upon loading 20 images, the scatterplots in Fig. 11 show strong biclusters in *Bison* and *Ox* classes and no clustering in other classes. Inspection of the *Ox* class images (via point selection in the scatterplot) reveals that one cluster predominantly contains Muskox samples, while another consists of Bull images. The separation between clusters suggests that the model distinguishes between two distinct visual subtypes within the *Ox* class, indicating intra-class ambiguity. We therefore partition this class into two subclasses: Muskox and Bull. Similarly, the *Bison* class can be divided into two previously unmodeled subclasses, corresponding to bison in cold and hot environments.

These subclass structures are also reflected in the layer's hierarchy view. Because the samples were selected as representative inputs for each class, the observed splits indicate that these ImageNet classes exhibit multimodal activation patterns, likely arising from overspecialization in certain model layers. Additional examples of subclassification are provided in the supplementary material.

Refining Model Classification. As part of the model refinement stage in the workflow shown in Fig. 3, we perform inference over the entire dataset to derive two additional subclasses: *Bison in hot & cold environments*, and *Bull* & *Muskox*. We then manually verify the alignment of these subclasses with their parent classes and applied corrections where necessary.

Subsequently, we retrain the model's final classification layer to accommodate 1002 classes. The updated model preserves the original accuracy and inference speed while increasing its classification capacity from 1000 to 1002 classes. This result demonstrates that the visualization-driven workflow, combined with limited human intervention, can effectively extend the model's representational granularity and classification capability. Further details of the transfer learning procedure are

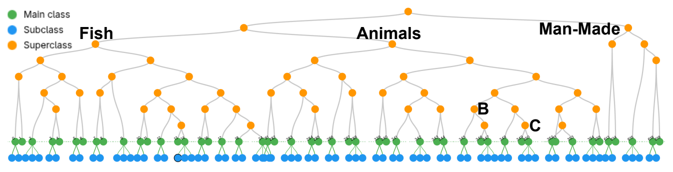

Fig. 10. A Confusion Hierarchy View constructed from ImageNet classes, showing automatic grouping into multiple super-classes at the mixed10 layer.

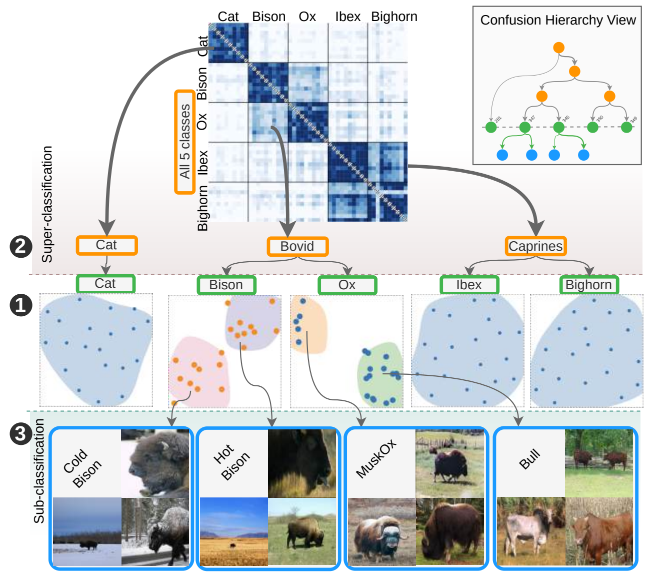

Fig. 11. An illustrated figure of the hierarchy view in *ChannelExplorer*. Starting from 5 classes **(1)**, we create 3 superclasses (*Cat*, *Bovid*, and *Caprines*) **(2)**. In the *Bison*'s Scatterplot View, two clusters show examples of *Bison*s in Cold & Hot weather **(3)**. Similarly for *Ox*, one cluster represents furry, barrel-shaped bodies (MuskOx breed), and the other represents less furry, cow-like bodies (Bull), leading to the creation of two subclasses. These (super/sub)classes can be linked with semantics by hovering and seeing the input images or hierarchy view nodes. Outline colors match those in the confusion hierarchy view.

provided in the supplementary material.

Class Relationships in GAN. Similar to the InceptionV3 experiment, we load images from the ImageNet classes *parachute*, *coral reef*, *cliff*, *orangutan*, and *persian cat* into SRResNet [68], a GAN architecture for super-resolution imaging. Although only the generator is used for inference, we analyze the internal representations of the discriminator to examine what it *perceives* when distinguishing between real high-resolution images and synthetically generated ones. In particular, we investigate whether inter-class relationships emerge during the discriminator's assessment of whether an input belongs to the distribution of authentic or generated images.

The Jaccard Similarity View of 5 representative images per user-defined class from the final convolutional block shows high similarities between *coral reef* & *cliff* and *orangutan* & *persian cat* (see Fig. 8). Although these similarities may appear unexpected given the absence of clear semantic relationships between the classes, they plausibly reflect the discriminator's reliance on low-level, texture-based cues rather than high-level semantic features. Furry textures in *orangutan* & *persian cat* and stone-like textures in *coral reef* & *cliff* may dominate during discriminator training, as the generator might imperfectly upscale them. The *parachute* class, mostly dominated by clear sky backgrounds, is added to standardize the colorscale of similarity matrix to improve visual clarity (similar to the use of *cat* class in InceptionV3).

Our visualization technique, applied with SRGAN, highlights how model inputs can show hierarchical relationships based on feature-level patterns, even if they lack semantic significance.

B. Scenario 2: Identification of Mislabeled Images

Following the workflow in Fig. 3, we begin by selecting the *Tiger Cat* class. The confusion hierarchy view indicates the presence of two subclasses. The Scatterplot View further reveals that a small subset of samples lies far from the main cluster. Inspecting these points shows that they correspond to images of *tigers*, suggesting potential mislabeling (see Fig. 12).

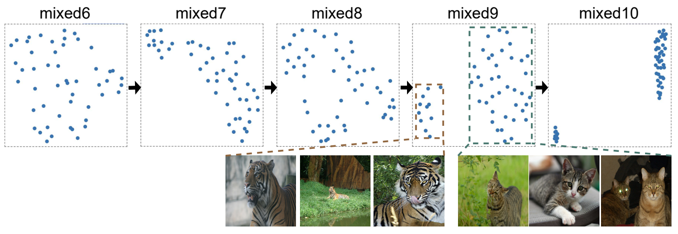

Fig. 12. Using 50 images from the *Tiger Cat* class, layers *mixed8* through *mixed10* exhibit consistent cluster formation. One cluster corresponds to images of the *Tiger* class, indicating potential mislabeling.

This separation behavior occurs in layers that identify features present in one input group (cat) but absent in another (tiger). As shown in Fig. 12, this distinction becomes noticeable after layer mixed9, where activation channels selectively activate for one group while deactivating for the other. We also find distinct stripe distributions between two groups from the same layer in the Heatmap View (see supplementary Sect. S2-C).

C. Scenario 3: Identify Activation Channels' Contribution

A channel is activated when it detects the presence of a feature. However, some detected features may be irrelevant to the target task (e.g., background patterns in an object detection setting), rendering the corresponding channels less informative. After loading inputs and analyzing class separability, we use *ChannelExplorer*'s Activation Heatmap View (step C in Figure 3) to identify such low-contribution channels.

For the mixed10 layer of the InceptionV3 model trained on ImageNet, we observe that channels at the bottom of the Heatmap View, ranked by the sum of class-pairwise distance, exhibit sporadic activations across all classes, indicating that they respond to non-discriminative features. Visual inspection shows that these channels often focus on border or corner regions of the input images rather than on class-relevant objects. Fig. 13 shows overlays of the last two rows of channels focusing around the border region.

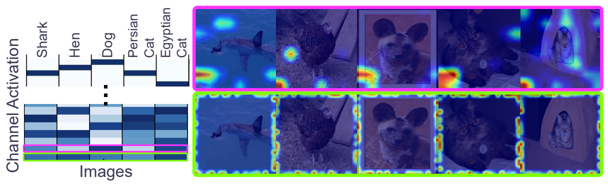

Fig. 13. Activation channels at the bottom of Heatmap View that identify unimportant features for the task. The first row always activates in the bottom left corner. The second row activates at the border regardless of the input image.

To evaluate the practical effect of removing such channels, we conduct channel-pruning experiments. The details on machine specification (CPU/GPU used) can be found in supplementary materials. We incrementally remove channels from the mixed10 layer in descending order of importance and re-evaluate the model on the 50,000-image ImageNet validation set. Each pruning configuration is tested 5 independent times, and we report the mean ± standard deviation across runs.

The baseline model achieves a Mean Average Precision (mAP) of 0.500 ± 0.003. Removing up to 74% of the channels preserves performance at 0.498 ± 0.004 mAP, indicating a negligible loss in accuracy. CPU inference time improves by 12.1%. GPU inference time remains unchanged until approximately 70% of channels are removed, after which it improves by about 50%. Memory consumption decreases linearly with the proportion of pruned channels on both CPU and GPU.

We compare this interpretable manual pruning approach with two established pruning strategies on VGG16: (1) unstructured magnitude-based weight pruning, and (2) structured filter pruning based on norm values.

Both baselines remove 40% of the layer weights. Their performance (0.501 ± 0.005 mAP) is comparable to that of our manually pruned model. However, our approach provides the additional advantage of visual, class-specific pruning guidance, allowing users to selectively adjust accuracy trade-offs across classes. Additional results on channel removal and detailed comparisons of the pruning methods are provided in the supplementary material.

D. Scenario 4: Locating Latent States' Position in SD

Understanding the latent position of the output of a model is crucial to determining the quality of the output. To determine the latent position of input in the Stable Diffusion (SD) model, we generate 100 text descriptions of different scenic places using LLM and focuse on the output of the *mid_block* layer of the U-Net model. To facilitate the visualization, we create user-defined classes, *Nature* and *Urban*, to the descriptions and select them in *ChannelExplorer* as step (A). The Scatterplot View in step (B) shows a clear separation between these two classes. Inspecting closely grouped images, we observe strong conceptual similarities (see Fig. 14).

If a newly generated image lies outside these clusters, we can assume that its features are different from those represented by the 100 prompts. We experiment with a prompt with intentionally ambiguous wording ("A large open space during day") that does not clearly imply either a natural or urban scene. The resulting image is positioned near the *Nature* cluster, suggesting that the model has a default preference toward interpreting underspecified prompts as natural environments. This example illustrates how *ChannelExplorer* can help identify such tendencies by positioning new generations in relation to an established latent distribution. However, we occasionally observe cases in which visually proximate points lack meaningful semantic similarity, underscoring the inherent limitations of projection-based views.

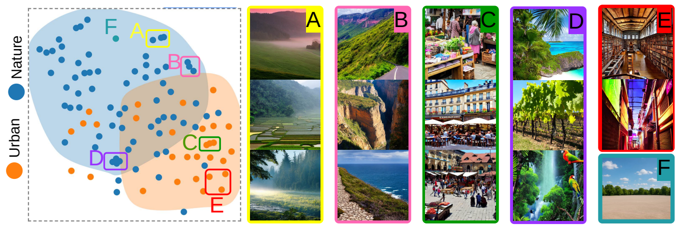

Fig. 14. Scatterplot View of the output from the *mid_block* layer of a Stable Diffusion model at the 49th timestep (out of 50), generated from 100 scene descriptions. Five point groups are highlighted on the right. Group A shares misty open-space characteristics; Group B features cliff-like mountainous landscapes; Group C contains dense crowd scenes; and Group D is characterized by large green trees. No strong common visual pattern is observed in Group E, aside from depicting urban environments. All input text prompts are provided in the supplementary material.

E. Computational and Visualization Scalability

Modern deep models can have hundreds of layers and millions of activations. We test the scalability of *ChannelExplorer* by loading multiple state-of-the-art models, such as Stable Diffusion ($\sim 860M$ parameters, 23 residual blocks) and EfficientNet ($\sim 66M$ parameters, 160+ image-based layers), which kept the system interactively responsive. We are able to load up to 200 inputs with 16GB RAM (memory usage scales linearly with number of inputs), demonstrating scalability across large datasets and architectures.

While the computational scalability of *ChannelExplorer* grows linearly with model and input size, visualization scalability is more constrained by human perception and harder to quantify. With representative input selection, up to 500 samples across all classes can be effectively visualized in the Dataset View, and both the Scatterplot View and Confusion Hierarchy View can display that many points without major visual clutter. However, dense matrix-based views such as the Jaccard Similarity and Heatmap views become less interpretable beyond ~ 50 inputs due to high cognitive load and limited screen space. Conversely, our experiments on multiple models demonstrate that channel activation distributions rarely exhibit more than four distinct modes (see example in Sect. S2-E showing two modes in activation distribution). Since the representative input selection samples one exemplar per activation cluster, selecting up to five examples per class provides sufficient diversity while mitigating the visualization scalability limits of the two views.

F. User Evaluation of ChannelExplorer

We performed a user evaluation to investigate whether participants can effectively use *ChannelExplorer* to **(a)** trace confusion across layers, **(b)** compare intra- and inter-class similarity, and **(c)** find misleading activation channels. In doing so, we sought to understand the tool's usefulness for hypothesis generation about model behavior, as well as its limitations. Due to the "black box" nature of deep learning models, there is no single correct outcome expected. Hence, our study focuses on whether users can consistently reason about model behavior using the tool and how the visual encodings support stable, interpretable decisions.

Our **controlled**, **within-subjects** evaluation sessions lasted one hour, and the **participants** were nine graduate students (P0-P8) who took a graduate Deep Learning course. P0-P3 are working with Deep Learning, P4-P5 are studying Computer Science, and P6-P8 are from other fields. The sessions took place in-person for P0, P2, P4, and P6 and virtually for others. One author functioned as the facilitator. In the user study, machine and display sizes are **independent variables**, success rate and time to reach first correct hypothesis are **dependent variables**, and the model, dataset, inputs are **controlled variables**. Further details on the study can be found at **osf.io/8zusf**.

The evaluation is done with InceptionV3 model and ImageNet dataset. Five images of five classes from Fig. 9 are used. Four tasks were performed during the evaluation.

T1: Load the dataset and guess four potentially confusing images. To create a hypothesis on class separability and test the effectiveness of user-defined class system, participants were asked to label and load the 25 images from Fig. 9 using the Dataset View. Without knowing model prediction, they guessed four images that might be confusing. This familiarized participants with the tool and later justified their assumptions. Participants were also asked several questions (e.g., input/output shapes, number of channels) to assess their understanding of CNNs and their familiarity with the tool. This evaluation measures the tool's learnability and the effectiveness of its initial onboarding process. All except P5 and P7 found the 1st *White Shark* and 4th *Persian Cat* images potentially confusable with the 3rd *Tiger Shark* and 3rd *Egyptian Cat*, respectively. However, P5 and P7 noted confusion between the 1st *Persian Cat* and 2nd *Egyptian Cat* images.

T2: Identify misleading layers for selected classes using Scatterplot View. Following an explanation of Scatterplot View, participants were asked to select confusing images from the scatterplot. P1-2 and P4-8 pointed to the 1st image of *White Shark* whereas P0 and P3 selected the 5th *Tiger Shark* and 5th *Persian Cat*, respectively. They were then asked to trace their selections backward, one preceding layer at a time, and identify the layer at which the confusion becomes less pronounced. This task directly evaluated the tool's capability of tracing confusions across layers (a). All participants identified a similar range of layers (mixed6–mixed10, except mixed8) as critical transition regions where separability changed unexpectedly. Furthermore, variations in DR initialization or cluster perceptions did not substantially impact participants' decisions on flagging a layer as the cause [69]. This suggests that *ChannelExplorer* supports stable layer-level reasoning even when low-level cluster embeddings vary. Participants hypothesized that retraining the identified layers may improve classification accuracy.

T3: Identify class confusion using Jaccard Similarity View. After introducing the Jaccard Similarity View, all participants agreed that *African Dog* exhibits the lowest level of confusion. Participants P0, P1, P3, P5, and P7 identified *Persian Cat* as having the highest intra-class confusion, whereas the remaining participants selected *White Shark*. All participants concurred that *White Shark* and *Tiger Shark* demonstrate the strongest inter-class confusion.

These findings suggest that the visualization consistently guides users toward the same high-confusion relationships, supporting our objective of evaluating the tool's effectiveness in identifying class similarity patterns (b).

T4: Identify misleading activation channels using Heatmap View. The participants were explained the activation channels' contribution and were asked to identify channels that activate wrongly for the classification from the Heatmap View. All participants responded with activation channels that are in the bottom half of the view. We removed all selected activation channels, and the model performance was unchanged.

This task demonstrates that *ChannelExplorer* enables users to reason about channel-level relevance (c) and supports hypothesis-driven pruning, even though the tool does not claim to pinpoint a single "faulty" channel.

Findings. Overall, the study provides qualitative evidence that *ChannelExplorer* supports the three intended analysis goals that the participants can consistently localize layers with low class separability, stably compare high and low inter- and intra-class confusions and identify misleading activation channels as pruning targets. Furthermore, variations of DR methods, K-means clustering, and cluster perception do not affect the decision to identify the target layers or activation channels. At the same time, the study highlights important limitations. Effective use of *ChannelExplorer* requires a solid understanding of neural networks. As intended, the tool highlights regions of interest rather than providing definitive diagnoses.

VII. Limitations of *ChannelExplorer*

ChannelExplorer overcomes the challenges in Sect. IV-A, however it has some limitations. The Heatmap View identifies less contributing activation channels for selected classes, but some of those channels might still be useful for other classes. Creating user-defined classes to analyze earlier layers in the network is challenging, as these layers capture basic features (shapes, textures, etc.) that are not task-related. While in Sect. VI-A, we show merging classes can improve model performance, some tasks may not allow changing the class numbers. Although any model can be loaded and visualized, all views work best on image-based layers (e.g., CNNs, transpose convolutions, residual blocks, pooling layers) due to the use of summarization functions. This limits applicability to other layer types like embedding layers, transformers, or attention layers. In advanced models like Stable Diffusion, which operate iteratively, the current system can only visualize a single user-defined iteration at a time. Comparing the same layer across iterations requires launching multiple tool instances. Finally, class separability remains a challenging and open problem. Like other state-of-the-art tools, *ChannelExplorer* cannot fully automate error diagnosis or implement corrective actions. Human interpretation and domain intuition remain essential components of the analysis process.

VIII. Discussion and Future Work

ChannelExplorer can be used and extended in several ways. While current system effectively reveals separability issues, identifying the contribution of each activation channel still relies on manual inspection. This process can be automated by integrating pattern recognition and vision-based models to interpret the semantics of activation maps, thereby reducing human involvement.

Furthermore, investigating neurons in terms of activation vectors can create alternative representation of model behavior as they may capture relational and directional properties between classes that are not apparent from activation channels alone. As transformer and attention-based architectures such as Vision Transformers (ViT) increasingly outperform CNNs in computer vision tasks [70], we plan to adapt our visual analytics methods to these models. Unlike convolutional activations that directly map to spatial features, attention representations reside in high-dimensional embedding spaces; visualizing and interpreting these latent interactions would broaden the applicability of *ChannelExplorer* beyond image-based layers.

Finally, our work suggests broader design implications for class separability analysis beyond image-based models. The workflow of progressing from abstract summaries to fine-grained internal representations, combined with comparative similarity views and linked multi-level exploration, is not inherently tied to images and is used in many other areas. These principles can be generalized to non-image-based models by replacing spatial activations with other meaningful internal representations (e.g., embedding vectors, attention weights, or neuron activations) and introducing new comparative visualization methods to support a hypothesis-driven analysis strategy.

IX. Conclusion

Image-based layers play a critical role in many neural network architectures for decision-making tasks, making their proper training and inference with the right dataset essential. Image-based layers are more interpretable and can be easily mapped on the inputs compared to other layer types (dense, recurrent, or dropout layers). For both beginners and experts, it is difficult to properly build a model without knowing how the model *see*s the dataset. Although there are tools for visualizing raw activations and high-level interpretation of CNN models, there is a gap in low-level explanations of the activation channels for any generic vision model.

In this paper, we present an image-based model visualization tool, *ChannelExplorer*, which is a data-driven visualization tool that helps debug the models and datasets at the level of activation channels. We have shown that the summarization methods and visualization techniques can help locate weaknesses and expose ways to improve them with minimal effort on a variety of architectures.

Acknowledgements

BW was partially supported by NSF grants DMS-2134223 and IIS-2205418.

References

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[2] P. Fraternali, F. Milani, R. N. Torres, and N. Zangrando, "Black-box error diagnosis in deep neural networks for computer vision: a survey of tools," *Neural Computing and Applications*, vol. 35, no. 4, pp. 3041–3062, 2023.

[3] D. Granziol, X. Wan, and T. Garipov, "Deep Curvature Suite," *arXiv preprint arXiv:1912.09656*, 2020.

[4] S. Horoi, J. Huang, G. Wolf, and S. Krishnaswamy, "Visualizing high-dimensional trajectories on the loss-landscape of ANNs," in *NeurIPS 2020 Workshop: Deep Learning through Information Geometry*, 2020.

[5] R. Bain, M. Tokarev, H. Kothari, and R. Damineni, "LossPlot: A better way to visualize loss landscapes," *arXiv preprint arXiv:2111.15133*, 2021.

[6] C. Seifert, A. Aamir, A. Balagopalan, D. Jain, A. Sharma, S. Grottel, and S. Gumhold, *Visualizations of Deep Neural Networks in Computer Vision: A Survey*. Springer International Publishing, 2017, pp. 123–144.

[7] L. Xue, X. Zhang, W. Jiang, K. Huo, and Q. Shen, "A classification performance evaluation measure considering data separability," in *Artificial Neural Networks and Machine Learning – ICANN*, 2023, pp. 1–13.

[8] D. Belcher, A. Prugel-Bennett, and S. Dasmahapatra, "Generalisation and the geometry of class separability," in *NeurIPS Workshop: Deep Learning through Information Geometry*, 2020.

[9] K. Ghosh, C. Bellinger, R. Corizzo, P. Branco, B. Krawczyk, and N. Japkowicz, "The class imbalance problem in deep learning," *Machine Learning*, vol. 113, no. 7, pp. 4845–4901, 2024.

[10] Y. Luo, Y. Wong, M. Kankanhalli, and Q. Zhao, "$\mathcal{G}$-Softmax: Improving intraclass compactness and interclass separability of features," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, pp. 685–699, 2020.

[11] A. Bilal, A. Jourabloo, M. Ye, X. Liu, and L. Ren, "Do convolutional neural networks learn class hierarchy?" *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 1, pp. 152–162, 2017.

[12] K. Cao, M. Liu, H. Su, J. Wu, J. Zhu, and S. Liu, "Analyzing the noise robustness of deep neural networks," *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 7, pp. 3289–3304, 2021.

[13] S. Carter, Z. Armstrong, L. Schubert, I. Johnson, and C. Olah, "Activation atlas," *Distill*, 2019.

[14] A. Karpathy, "t-SNE visualization of CNN codes," https://cs.stanford.edu/people/karpathy/cnnembed/.

[15] D. Bau, B. Zhou, A. Khosla, A. Oliva, and A. Torralba, "Network dissection: Quantifying interpretability of deep visual representations," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 3319–3327.

[16] R. Fong and A. Vedaldi, "Net2Vec: Quantifying and explaining how concepts are encoded by filters in deep neural networks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 8730–8738.

[17] B. Kim, M. Wattenberg, J. Gilmer, C. Cai, J. Wexler, F. Viegas, and R. Sayres, "Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (TCAV)," in *International Conference on Machine Learning*, 2018, pp. 2668–2677.

[18] M. Rahat-Uz-Zaman, S. Hye, and M. A. H. Akhand, "Start-end writing integration with convolutional neural network for Bengali handwritten numeral recognition," in *International Conference on Electrical and Computer Engineering (ICECE)*, 2022, pp. 348–351.

[19] F. Hohman, H. Park, C. Robinson, and D. H. Chau, "Summit: Scaling deep learning interpretability by visualizing activation and attribution summarizations," *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 1, pp. 1096–1106, 2020.

[20] C. Olah, A. Satyanarayan, I. Johnson, S. Carter, L. Schubert, K. Ye, and A. Mordvintsev, "The building blocks of interpretability," *Distill*, 2018.

[21] G. Goh, N. Cammarata, C. Voss, S. Carter, M. Petrov, L. Schubert, A. Radford, and C. Olah, "Multimodal neurons in artificial neural networks," *Distill*, 2021.

[22] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell, "Deep domain confusion: Maximizing for domain invariance," *arXiv preprint arXiv.1412.3474*, 2014.

[23] D. Erhan, Y. Bengio, A. Courville, and P. Vincent, "Visualizing higher-layer features of a deep network," *Technical Report, University of Montreal*, 2009.

[24] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson, "Understanding neural networks through deep visualization," in *Deep Learning Workshop, International Conference on Machine Learning (ICML)*, 2015.

[25] C. Olah, A. Mordvintsev, and L. Schubert, "Feature visualization," *Distill*, 2017.

[26] A. Hinterreiter, P. Ruch, H. Stitz, M. Ennemoser, J. Bernard, H. Strobelt, and M. Streit, "ConfusionFlow: A model-agnostic visualization for temporal analysis of classifier confusion," *IEEE Transactions on Visualization and Computer Graphics*, vol. 28, no. 2, pp. 1222–1236, 2022.

[27] M. Pühringer, A. Hinterreiter, and M. Streit, "InstanceFlow: Visualizing the evolution of classifier confusion at the instance level," in *2020 IEEE Visualization Conference (VIS)*, 2020, pp. 291–295.

[28] H. Zeng, H. Haleem, X. Plantaz, N. Cao, and H. Qu, "CNNComparator: Comparative analytics of convolutional neural networks," *arXiv preprint arXiv:1710.05285*, 2017.

[29] J. Talbot, B. Lee, A. Kapoor, and D. S. Tan, "EnsembleMatrix: interactive visualization to support machine learning with multiple classifiers," in *ACM SIGCHI Conference on Human Factors in Computing Systems*, 2009, pp. 1283–1292.

[30] J. Görtler, F. Hohman, D. Moritz, K. Wongsuphasawat, D. Ren, R. Nair, M. Kirchner, and K. Patel, "Neo: Generalizing confusion matrix visualization to hierarchical and multi-output labels," in *ACM SIGCHI Conference on Human Factors in Computing Systems*, 2022, pp. 1–13.

[31] J. Wang, L. Gou, H. Shen, and H. Yang, "DQNViz: A visual analytics approach to understand deep q-networks," *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 1, pp. 288–298, 2019.

[32] S. Chung, S. Suh, C. Park, K. Kang, J. Choo, and B. C. Kwon, "ReVACNN: Real-time visual analytics for Convolutional Neural Network," *ACM SIGKDD Workshop on Interactive Data Exploration and Analytics (IDEA)*, 2016.

[33] D. Liu, W. Cui, K. Jin, Y. Guo, and H. Qu, "DeepTracker: Visualizing the training process of convolutional neural networks," *ACM Transactions on Intelligent Systems and Technology*, vol. 10, no. 1, pp. 1–25, 2019.

[34] A. Kapoor, B. Lee, D. Tan, and E. Horvitz, "Interactive optimization for steering machine classification," in *ACM SIGCHI Conference on Human Factors in Computing Systems*, 2010, pp. 1343–1352.

[35] Y. Abraham and N. Sauwen, "Radviz: Project multidimensional data in 2d space," https://CRAN.R-project.org/package=Radviz, 2025.

[36] B. Alsallakh, A. Hanbury, H. Hauser, S. Miksch, and A. Rauber, "Visual methods for analyzing probabilistic classification data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 12, pp. 1703–1712, 2014.

[37] D. Ren, S. Amershi, B. Lee, J. Suh, and J. D. Williams, "Squares: Supporting interactive performance analysis for multiclass classifiers," *IEEE Transactions on Visualization & Computer Graphics*, vol. 23, no. 1, pp. 61–70, 2017.

[38] H. Hoeiness, A. Harstad, and G. Friedland, "From tinkering to engineering: Measurements in Tensorflow Playground," *arXiv preprint arXiv.2101.04141*, 2021.

[39] J. Bian, "Quiver," https://github.com/keplr-io/quiver, 2016.

[40] A. Ghorbani, J. Wexler, J. Zou, and B. Kim, "Towards automatic concept-based explanations," in *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, 2019, pp. 9273–9282.

[41] J. Huang, A. Mishra, B. C. Kwon, and C. Bryan, "ConceptExplainer: Interactive explanation for deep neural networks from a concept perspective," *IEEE Transactions on Visualization and Computer Graphics*, vol. 29, no. 1, pp. 831–841, 2023.

[42] M. Kahng, P. Y. Andrews, A. Kalro, and D. H. P. Chau, "ActiVis: Visual exploration of industry-scale deep neural network models," *IEEE Transactions on Visualization & Computer Graphics*, vol. 24, no. 1, pp. 88–97, 2018.

[43] N. Pezzotti, T. Höllt, B. Lelieveldt, E. Eisemann, and A. Vilanova, "Hierarchical Stochastic Neighbor Embedding," *Computer Graphics Forum*, vol. 35, no. 3, pp. 21–30, 2016.
[44] N. Pezzotti, T. Höllt, J. Van Gemert, B. P. Lelieveldt, E. Eisemann, and A. Vilanova, "DeepEyes: Progressive visual analytics for designing deep neural networks," *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 1, pp. 98–108, 2017.
[45] A. Nguyen, J. Clune, Y. Bengio, A. Dosovitskiy, and J. Yosinski, "Plug & Play generative networks: Conditional iterative generation of images in latent space," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 3510–3520.
[46] A. Mordvintsev, C. Olah, and M. Tyka, "Inceptionism: Going deeper into neural networks," https://research.google/blog/inceptionism-going-deeper-into-neural-networks/, 2015.
[47] OpenAI, "OpenAI Microscope," https://openai.com/index/microscope/.
[48] N. Das, H. Park, Z. J. Wang, F. Hohman, R. Firstman, E. Rogers, and D. H. P. Chau, "Bluff: Interactively deciphering adversarial attacks on deep neural networks," in *2020 IEEE Visualization Conference (VIS)*, 2020, pp. 271–275.
[49] C. Molnar, *Interpretable Machine Learning: A Guide for Making Black Box Models Explainable*, 3rd ed., 2025. [Online]. Available: https://christophm.github.io/interpretable-ml-book/
[50] Z. J. Wang, R. Turko, O. Shaikh, H. Park, N. Das, F. Hohman, M. Kahng, and D. H. Chau, "CNN Explainer: Learning convolutional neural networks with interactive visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 2, pp. 1396–1406, 2020.
[51] A. Cho, G. C. Kim, A. Karpekov, A. Helbling, Z. J. Wang, S. Lee, B. Hoover, and D. H. P. Chau, "TRANSFORMER EXPLAINER: Interactive learning of text-generative models," in *Proceedings of the Thirty-Ninth AAAI Conference on Artificial Intelligence and Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence and Fifteenth Symposium on Educational Advances in Artificial Intelligence*, 2025, pp. 29 625–29 627.
[52] S. Lee, B. Hoover, H. Strobelt, Z. J. Wang, S. Peng, A. Wright, K. Li, H. Park, H. Yang, and D. H. P. Chau, "Diffusion Explainer: Visual explanation for text-to-image stable diffusion," in *2024 IEEE Visualization and Visual Analytics (VIS)*, 2024, pp. 96–100.
[53] M. Kahng, N. Thorat, D. H. Chau, F. B. Viégas, and M. Wattenberg, "GAN Lab: Understanding complex deep generative models using interactive visual experimentation," *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 1, pp. 310–320, 2018.
[54] M. Bellgardt, C. Scheiderer, and T. W. Kuhlen, "An immersive node-link visualization of artificial neural networks for machine learning experts," in *2020 IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR)*, 2020, pp. 33–36.
[55] A. W. Harley, "An interactive node-link visualization of convolutional neural networks," in *Advances in Visual Computing*, G. Bebis, R. Boyle, B. Parvin, D. Koracin, I. Pavlidis, R. Feris, T. McGraw, M. Elendt, R. Kopper, E. Ragan, Z. Ye, and G. Weber, Eds. Cham: Springer International Publishing, 2015, pp. 867–877.
[56] S. P. Leeman-Munk, S. Sethi, C. G. Healey, S. Nie, K. Padia, R. Devarajan, D. J. Caira, J. R. Benson, J. A. Cox, L. E. L. Lewis, and M. O. Kabul, "Visualizing convolutional neural networks," https://patents.google.com/patent/US10192001B2, 2019.
[57] C. Schorr, P. Goodarzi, F. Chen, and T. Dahmen, "Neuroscope: An explainable ai toolbox for semantic segmentation and image classification of convolutional neural nets," *Applied Sciences*, vol. 11, no. 5, 2021.
[58] A. Rathore, N. Chalapathi, S. Palande, and B. Wang, "TopoAct: Visually exploring the shape of activations in deep learning," *Computer Graphics Forum*, vol. 40, no. 1, pp. 382–397, 2021.
[59] D. E. Jacobs, D. B. Goldman, and E. Shechtman, "Cosaliency: where people look when comparing images," in *ACM Symposium on User Interface Software and Technology (UIST)*, 2010, pp. 219–228.
[60] A. Rosenfeld, M. D. Solbach, and J. K. Tsotsos, "Totally Looks Like - how humans compare, compared to machines," in *IEEE Computer Vision and Pattern Recognition Workshops*, 2018, pp. 1961–1964.
[61] M. Eiglsperger, M. Siebenhaller, and M. Kaufmann, "An efficient implementation of Sugiyama's algorithm for layered graph drawing," in *Graph Drawing*, 2005, pp. 155–166.
[62] B. Shneiderman, "The eyes have it: a task by data type taxonomy for information visualizations," in *Proceedings 1996 IEEE Symposium on Visual Languages*, 1996, pp. 336–343.
[63] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.
[64] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft, "When is "nearest neighbor" meaningful?" in *Database Theory — ICDT*, 1999.
[65] D. Pelleg and A. W. Moore, "X-means: Extending K-means with efficient estimation of the number of clusters," in *International Conference on Machine Learning (ICML)*, 2000, pp. 727–734.
[66] E. Catmull and R. Rom, "A class of local interpolating splines," in *Computer Aided Geometric Design*, 1974.
[67] B. O. Ayinde and J. M. Zurada, "Building efficient ConvNets using redundant feature pruning," *arXiv preprint arXiv.1802.07653*, 2018.
[68] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, "Photo-realistic single image super-resolution using a generative adversarial network," in *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 105–114.
[69] H. Jeon, G. J. Quadri, H. Lee, P. Rosen, D. A. Szafir, and J. Seo, "CLAMS: A cluster ambiguity measure for estimating perceptual variability in visual clustering," *IEEE Transactions on Visualization and Computer Graphics*, vol. 30, no. 1, pp. 770–780, 2024.
[70] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations*, 2021.
[71] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, "A ConvNet for the 2020s," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 11 976–11 986.

Md Rahat-uz- Zaman is a PhD student at the Scientific Computing and Imaging (SCI) Institute and the Kahlert School of Computing, University of Utah. His research focuses on visualization and explainable AI, with an emphasis on generalized visual analytics tools to improve the interpretability of deep learning models.

Bei Wang is an Associate Professor in the School of Computing and a faculty member of the Scientific Computing and Imaging (SCI) Institute at the University of Utah. She received the PhD degree in computer science from Duke University. Her research interests include topological data analysis, scientific and information visualization, computational and applied topology, computational geometry, and machine learning. She is the recipient of a U.S. Department of Energy Early Career Research Program Award (2020), a National Science Foundation CAREER Award (2022), and the Presidential Early Career Award for Scientists and Engineers (PECASE) (2024).

Paul Rosen received a PhD degree from Computer Science Department, Purdue University. He is an associate professor with the University of Utah. His research interests include applying geometry– and topology-based approaches to problems in information visualization. Along with his collaborators, he has received best paper awards or honorable mentions at IEEE VIS, IEEE Pacific Vis, CG&A, IVAPP, and SIBGRAPI. He received a National Science Foundation CAREER Award, in 2019.