Article Type: Description (see Introduction for more detail)

MuCHEx: A Multimodal Conversational Debugging Tool for Interactive Visual Exploration of Hierarchical Object Classification

Reza Shahriari, *University of Florida, Gainesville, FL, 32608, USA*

Yichi Yang, *University of California, San Diego, La Jolla, CA, 92093, USA*

Danish Nisar Ahmed Tamboli, *University of Florida, Gainesville, FL, 32608, USA*

Michael Perez, *University of Florida, Gainesville, FL, 32608, USA*

Yuheng Zha, *University of California, San Diego, La Jolla, CA, 92093, USA*

Jinyu Hou, *Carnegie Mellon University, Pittsburgh, PA, 15213, USA*

Mingkai Deng, *Carnegie Mellon University, Pittsburgh, PA, 15213, USA*

Eric D. Ragan, *University of Florida, Gainesville, FL, 32608, USA*

Jaime Ruiz, *University of Florida, Gainesville, FL, 32608, USA*

Daisy Zhe Wang, *University of Florida, Gainesville, FL, 32608, USA*

Zhitting Hu, *University of California, San Diego, La Jolla, CA, 92093, USA*

Eric Xing, *Carnegie Mellon University, Pittsburgh, PA, 15213, USA*

Abstract—Object recognition is a fundamental challenge in computer vision, particularly for fine-grained object classification, where classes differ in minor features. Improved fine-grained object classification requires a teaching system with numerous classes and instances of data. As the number of hierarchical levels and instances grows, debugging these models becomes increasingly complex. Moreover, different types of debugging tasks require varying approaches, explanations, and levels of detail. We present MuCHEx, a multimodal conversational system that blends natural language and visual interaction for interactive debugging of hierarchical object classification. Natural language allows users to flexibly express high-level questions or debugging goals without needing to navigate complex interfaces, while adaptive explanations surface only the most relevant visual or textual details based on the user's current task. This multimodal approach combines the expressiveness of language with the precision of direct manipulation, enabling context-aware exploration during model debugging.

Object recognition is a core challenge in computer vision, encompassing two main tasks: object instance recognition and object class recognition [1]. However, most current systems struggle with object class recognition, like distinguishing between specific breeds of animals rather than general species, or identifying different models of vehicles rather than just recognizing them as cars or trucks. This complexity arises because real-world visual categories follow natural hierarchies with multiple levels of detail.

Distinguishing between different classes, especially

XXXX-XXX © 2025 IEEE
Digital Object Identifier 10.1109/XXX.0000.0000000

with only limited visual examples, can be challenging even for humans. This is especially tough in fine-grained hierarchical classification, where classes differ in minor features [2]. Models trained on limited examples—common in few-shot learning scenarios—often struggle with sparse data at deeper levels of the hierarchy. Improved fine-grained object classification requires a teaching system with numerous classes and instances of data. Despite recent improvements in methods and training strategies, these few-shot learning models are not perfect. Fixing issues related to misclassification at fine-grained hierarchical levels before deployment is critical, as overlooked errors can propagate misleading outputs. For instance, misidentifying medical imagery or incorrectly categorizing safety-critical equipment could lead to significant practical consequences.

As the number of classes and instances grows, the space of possible mistakes and explanations also expands rapidly. Each class can be confused with many others, and each instance might have unique features or edge cases that require individualized attention. This combinatorial explosion makes it harder to systematically explore, identify, and correct errors, leading to a much more complex and time-consuming debugging process. Analysts must be able to query and review both high-level classifications and low-level segmentations or attributes, often across multiple examples. This creates a demand for flexible, task-driven exploration where users can pursue different types of inquiry and explanation depending on the task at hand. For example, while correcting a misclassification may only require reviewing accuracy scores and alternative label candidates, understanding why the misclassification occurred demands deeper details, such as identifying which essential attributes or segments were present, missing, or misinterpreted in the instance. Thus, our goal is to provide multi-scope explanations and representations to support different task types that help debuggers identify and understand error patterns.

Debugging classification models is inherently challenging because different types of errors often require different approaches and levels of detail. Solely textual outputs can limit a user's ability to efficiently locate errors in visual data, as they lack spatial context or visual cues. However, purely visual interactions may be ambiguous or insufficient for conveying underlying statistical patterns or model rationales, especially when precise numeric explanations are required. Natural language input supports more natural communication by allowing users to articulate debugging questions and reasoning goals in their own words, while direct manipulation offers precision and control—together enabling more effective and flexible interaction during classification debugging tasks [3]. In this paper, we present a *multimodal* tool for exploratory debugging hierarchical object classification that supports both textual and visual output, and enables dual-mode interaction: (1) *natural language interaction* for conversational querying, and (2) *direct manipulation* of visual elements such as image segments and concept hierarchies. The interface (see Figure 1) integrates multiple coordinated views: (A) a natural language chat interface, (B) a concept visualization panel, and (C) image-based interaction—to enable *adaptive, task-driven exploration* of model behavior.

This paper makes two primary contributions: (1) We introduce MuCHEx, a *Multimodal Conversational Debugging Tool for Interactive Visual Exploration of Hierarchical Object Classification*, which integrates coordinated views, natural language interaction, and adaptive visual explanations to support model understanding and error analysis; (2) We present the results of a human-subjects study evaluating MuCHE's effectiveness in helping users identify and correct different types of classification errors in a few-shot learning scenario.

DESIGN GOALS

This research focuses on debugging hierarchical object classification due to the challenges and similarities of real-world objects along especially with limited examples seen by models. The presented application combines multiple techniques to provide explanations for model analysis and debugging of various types of model issues. Explainable AI (XAI) strategies seek to enhance transparency in a model's decision-making process. These explanations can help identify and find the reason when the model makes errors [4]. Explanations for hierarchical cases involve the challenge of explaining relationships at different levels of a hierarchy. We can consider animal classification as an example; explaining animal differences can extend to families, species, and subspecies. For instance, distinguishing between African elephants and Asian elephants—or even more specific contrasts between African savannah and African forest elephants—might be challenging for humans. Identifying and correcting model errors would be even harder without the ability to inspect or communicate the necessary differences up or down a branch in the hierarchy, or it may involve comparisons across multiple branches. We define a set of three interdependent design goals that guide the development of our interactive system, derived through analysis of prior work [4], [5], [6], [7]. Ex-

This article has been accepted for publication in IEEE Computer Graphics and Applications. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/MCG.2025.3598204

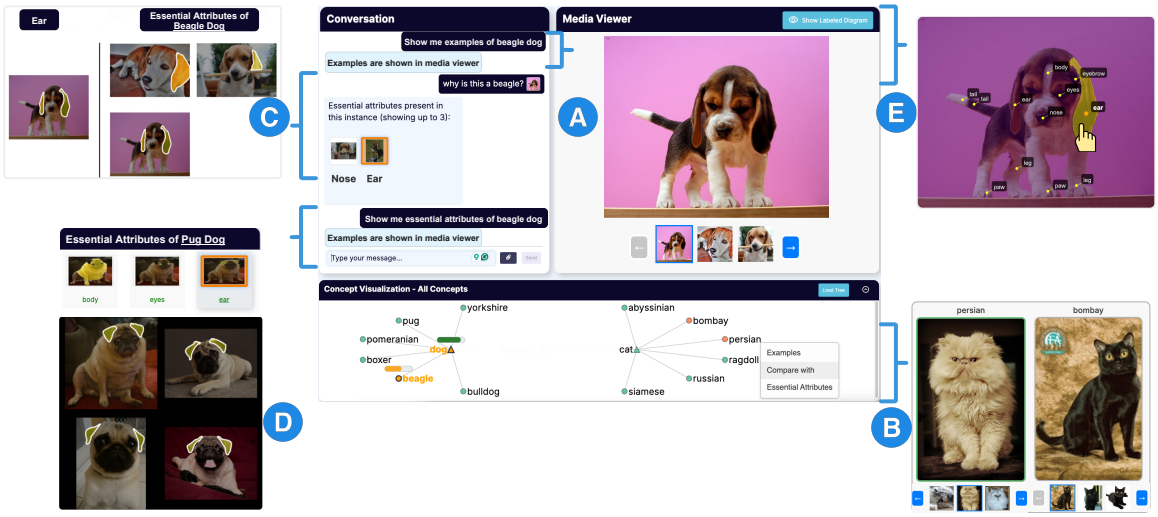

FIGURE 1. Screenshots from the interface after executing user commands: (A) Examples Viewer mode: displays representative images of the predicted class; (B) Comparison Viewer mode: shows side-by-side contrasts between candidate classes to support comparison; (C) Reasoning visual annotation: highlights visual evidence used by the model to support or reject predictions; (D) Essential Attributes mode: displays key part-level segments that are considered necessary for classification; (E) Labeled Diagram: overlays part names directly on the image to aid verifying segmentations

ploring classification behavior in hierarchical models requires users to navigate between global overviews of class structures and detailed, instance-level information; managing these levels simultaneously can be cognitively demanding and overwhelming. Coordinating them within a unified interface reduces this load and improves task performance [5].

Furthermore, users engage in a variety of debugging tasks, from querying model decisions to inspecting visual features that benefit from flexible forms of input. Relying solely on either natural language or visual interaction restricts expressiveness and control, whereas combining both modalities enables users to articulate goals conversationally and manipulate visual elements directly, improving efficiency and reducing cognitive effort [6]. Finally, presenting all possible explanations at once can overload users and hinder effective analysis; adaptively surfacing only context-relevant information helps maintain user focus and supports better decision-making during model exploration [7].

DG1. Supporting Multi-Level Exploration of Instance Details and Class Hierarchies As classification models scale across hundreds of classes and hierarchical levels, understanding why a particular label was assigned demands integrating both global (overview of different class hierarchies, such as the breakdown of animals into species) and local (instance-specific, such as images of a boxer dog) views. A key challenge is that users must juggle both global-level details, such as navigating class hierarchies, and instance-level information like label alternatives and confidence scores for selected images. This often requires multiple steps across separate views to find the details needed, increasing cognitive effort and making it easy to lose track of previously examined details. To address this, our design leverages multiple coordinated views that synchronize explanations across different levels and instances, and support interactive inspection of data features, highlighting the predicted class in the hierarchy, displaying confidence levels, and generating explanatory text in the chat interface. Studies of multiple coordinated views demonstrate that linked views improve task performance, reveal unforeseen relationships, and reduce cognitive load when exploring complex hierarchies [5]. Embedding these coordination mechanisms in an XAI tool helps users detect errors across large datasets.

In information visualization, annotations, such as labels, highlights, or explanatory text, improve comprehension, engagement, and memory retention. When tailored to specific analysis phases (e.g., data cleaning, exploration, or error detection), they further enhance users' ability to identify and understand issues [8]. In our setting, users often face challenges interpreting segment-level labels across multiple instances, partic-

ularly when model predictions are fine-grained or ambiguous. Because the model's reasoning relies heavily on image segmentation to identify class-defining regions, it is crucial that users can assess the accuracy and alignment of these segments with semantic concepts to verify and debug predictions. Also, we provide explanations at the instance level by linking multiple views: an overview radial tree, a detail-view image panel, conversational chat box. Selection, filter, or highlight in one view automatically propagates to the others, letting users navigate between individual edge cases (e.g., one misclassified image). This coordination reduces context-switching and enables users to locate and interpret model decisions from multiple views immediately, as can be seen in Figure 1.

DG2. Enable Dual-Modal Interaction for Flexible Exploration To support complex reasoning tasks during model debugging, this design goal emphasizes the use of dual-modal interaction by incorporating both natural language (NL) and visual manipulation as input modalities, and delivering both textual and visual forms of output. This duality ensures that users can flexibly express their goals and receive explanations in the mode best suited to the task at hand. For example, users who are unfamiliar with the system may prefer to begin with natural language input (e.g., asking "Why is this a boxer dog?"), allowing them to receive visual explanations such as highlighted segments or hierarchy paths. In contrast, when users are dealing directly with visual entities, such as image segments or classification nodes, they are more likely to manipulate visual elements directly such. Also, users prefer multimodal interfaces over unimodal ones because they combine the expressive freedom of natural-language input with the precision and immediacy of direct manipulation, due to their support of the freedom of natural language expression and complementary nature of direct manipulation [6]. We integrate direct manipulation—allowing users to click, and receive immediate, reversible visual feedback—with an embedded natural-language interface for conversational queries (e.g., "Why is this a boxer dog?" or "Compare beagle and boxer dogs.") as can be seen in Figure 1 C and Figure 1 B. This combination lowers cognitive effort, accommodates users with different expertise levels, and lets debuggers seamlessly switch between exploratory GUI-based interaction and goal-driven questioning to debug hierarchical classifications more efficiently.

DG3. Task-driven Debugging through Adaptive Information Views This design goal focuses on supporting dynamic adaptation of views based on the user's current task and mode of interaction. Different interactions, such as posing a natural language question, clicking a segment, or selecting a class node indicate different explanatory needs. Because the system contains many layers of detail (e.g., class hierarchies, confidence scores, segmentation masks, etc.), presenting all information at once would be overwhelming for users Instead, the system dynamically surfaces only the most relevant explanations for the current interaction, such as highlighting the class path when asking "*What is this*?", or visualizing attribute-level differences when comparing classes while allowing users to explore additional details on demand. Selecting dynamic and adaptive views that follow necessary features maximizes the amount of information that can be perceived throughout a sequence and improves user task performance [7].

MODEL

Our concept learning system adopts a two-stage process to learn object concepts. In the first stage, it identifies common part concepts across the dataset and organizes them into a hierarchy. In the second stage, it treats this part hierarchy as a concept bottleneck layer [9] and learns the association between parts and object-level concepts.

We choose to model part concepts as a hierarchy since part concepts of different granularities are useful for composing different types of object concepts. For example, while the generic part concept "wheel" is strongly associated with the concept "land vehicle", a more granular part "train wheel" is needed to represent the more specific object "train". Given a set of object images $\{x_1, x_2, \ldots, x_N\}$, to learn this part hierarchy in an unsupervised fashion, we first decompose each image x_i into segmented parts $\{x_i^{(1)}, x_i^{(2)}, \ldots, x_i^{(M_i)}\}$ using a segmentation model. Next, we apply an online hierarchical agglomerative clustering algorithm [10] to cluster all part segments into a binary tree T based on the DINO v2 representation [11] of the part segments. The leaves of T are $\bigcup_{i=1}^{N}\{x_i^{(1)}, x_i^{(2)}, \ldots, x_i^{(M_i)}\}$, all part segments seen so far. Each subtree of T represents a set of semantically similar part segments (e.g., different kinds of wheels), which can be interpreted as a part concept. This results in a rich set of hierarchical parts concepts that we can compose to define object concepts.

In the second stage, the system learns how part concepts are composed to form objects by fitting a logistic regression classifier. Given the set of labeled images $\{x_1, x_2, \ldots, x_N\}$ from $|C|$ object categories, their category labels $\{y_1, y_2, \ldots, y_N\}$, and the part hierarchy T learned in the first stage, we first featurize each image as a $|T|$ dimensional binary vector $z_i \in \{0,1\}^{|T|}$

This article has been accepted for publication in IEEE Computer Graphics and Applications. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/MCG.2025.3598204

based on if it contains part of each subtree in the hierarchy. Then, following [12], [13], we fit a logistic regression model to predict the labels y_i with strong ElasticNet regularization [14].

Concretely, given learnable weights $W_\phi \in \mathbb{R}^{|C|\times|T|}$, regularization strength λ, and ElasticNet ratio α, we train the model to optimize the following objective:

$$\min_{W_\phi} \frac{1}{N}\sum_{i=1}^{N} \text{CrossEntropyLoss}(W_\phi z_i, y_i) + \lambda\, \text{R}(W_\phi)$$

$$\text{where } \text{R}(W_\phi) = \frac{1}{2}(1-\alpha)||W_\phi||_2^2 + \alpha||W_\phi||_1$$

$$\text{s.t. } W_\phi \geq 0\ .$$

As strong regularization results in a sparse weight matrix, and the non-negativity constraint restricts non-zero entries to be positive, W_ϕ can be easily interpreted as representing each object concept (rows) as the combination of a small set of part concepts (positive columns).

We trained and evaluated the system on the Cars [15], FGVC-Aircraft [16], and Oxford-IIIT Pet [17] datasets. Images in these datasets are labeled with hierarchical labels (e.g., images in Cars are annotated with *make* and *model*). To support hierarchal classification, using the above objective, we trained one logistic regression classifier for each parent node in the hierarchy [18]. For instance, for the Cars dataset, we trained one classifier to predict the *make*, and one classifier per unique *make* to predict its *model*s. This decomposes hierarchal classification into a series of multiclass classification problems and allows us to learn the association between object and part concepts at different levels of the hierarchy.

We evaluated the model in a few-shot learning setting, where we sample *n* instances for each lowest level class to train the model, and then test the model on the test split. The results are shown in Table 1. The results show that higher-level classifications (e.g., *Make* or *Type*) consistently achieve higher accuracy, even in the 1-shot setting—for example, Oxford's *Type* reaches over 94% with just one example. In contrast, lower-level categories like *Model*, *Variant*, and *Breed* have much lower 1-shot accuracy, highlighting the increased difficulty of fine-grained classification. However, accuracy for these fine-grained levels improves substantially with more examples, demonstrating the benefit of few-shot learning in reducing the performance gap.

To guide the system's design and evaluation, we conducted a preliminary error analysis of the model outputs across several datasets mentioned in few-shot settings. Our analysis showed that errors typically fell into three distinct categories: (1) *classification errors*, where the predicted label was incorrect at either the coarse or fine-grained level; (2) *segmentation errors*, where the system failed to correctly identify or locate the key parts used for classification; and (3) *essential attribute errors*, where the model incorrectly learned spurious correlations (for example, treating background artifacts as essential features).

TABLE 1. Hierarchical classification accuracy with few shot training across multiple datasets and different training shots (*n*). Accuracy values presented as percentages. As expected, accuracy improves with more training examples, highlighting the benefits of few-shot learning. Higher-level categories (e.g., type or make) achieve relatively strong performance even with limited data, while fine-grained classes (e.g., breed or model) remain challenging, underscoring the difficulty of multiclass, hierarchical classification.

Dataset	*Level*	*Class Count*	*Accuracy* (%)		
			1-shot	4-shot	16-shot
Cars [15]	Make	49	32.3	61.4	82.8
	Model	196	12.9	36.5	65.5
FGVC Aircraft [16]	Make	30	40.2	58.6	77.9
	Model	70	23.5	41.6	63.2
	Variant	100	14.9	29.3	48.7
Oxford IIIT Pet [17]	Type	2	94.2	99.1	99.3
	Breed	37	23.4	52.5	68.6

SYSTEM DESIGN

To address our design challenges, we propose MuCHEx, a multimodal tool that supports detecting the reasoning of various explanations for object classification. It enables users to fluidly navigate between visual inspection and natural language dialogue, with tailored responses that adapt to the selected element, task type, or query. The web-based interface for MuCHEx is implemented using HTML/CSS, Bootstrap (for styling and responsive layout), jQuery, and D3.js (for all interactive visualizations).

Conversational Chat Box

The chat box serves as the natural-language interface for the entire system. Users pose questions or commands in free text (e.g., "*Why is this a boxer dog?*"), and the backend replies in threaded dialogue. These responses are context-aware and visually enriched, incorporating elements like labeled thumbnails, color-coded highlights, and ranked candidate labels to show relevant evidence. This design turns each chat exchange into an entry point, where explanations are not only textual but grounded in interactive visual elements that the user can manipulate further. Depending on

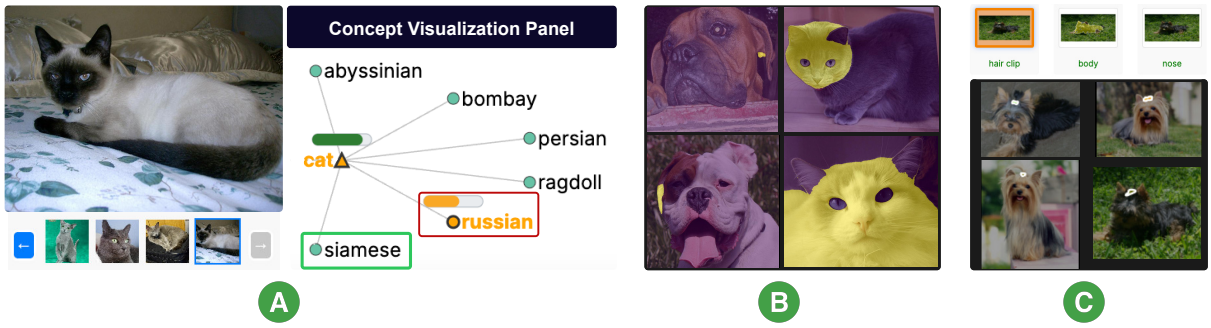

FIGURE 2. Screenshots illustrating different types of model errors investigated in the evaluation study. A *Classification error*: This instance is a *Siamese* cat, but the system incorrectly classified it as a *Russian* with low confidence. B *Segmentation error*: The system failed to detect the ear in the left column and the nose in the right column. C *Incorrect essential attributes*: The system incorrectly identifies a hair clip as essential for this breed, likely due to it appearing in all training examples, even though it is not semantically relevant.

the query type, the system dynamically adjusts the response format (*DG3*). For example, using side-by-side thumbnails to explain fine-grained distinctions as shown in Figure 1 C The inclusion of text labels on images supports faster interpretation and helps users connect linguistic and visual information with minimal effort. On demand, they can click to see the full details if needed (*DG1*). This couples language with imagery, letting users transition fluidly between descriptive reasoning ("*compare examples of...*") and exploratory follow-ups triggered by a single click on an embedded thumbnail (*DG2*).

Moreover, a key feature of the chat interface is the use of the keyword "*this*," which refers to the currently selected object or part within the Media Viewer. When users ask follow-up questions like "*Why is this a beagle?*" or "*What is this?*", the system anchors the query to the specific visual element under inspection, displaying a thumbnail next to the message for clarity. This explicit linking between language and imagery establishes strong multimodal coordination and ensures users never lose visual context during conversation-driven interactions.

In addition, our system supports a diverse set of natural-language commands, which can be grouped into several categories:

- *Descriptive queries* (e.g., "*What is this?*", "*This is a NAME.*") prompt the system to generate explanations with candidate labels and confidence levels.
- *Comparative commands* (e.g., "*Compare examples of a beagle with a boxer*") trigger side-by-side image displays in the Media Viewer and highlight distinguishing features in the Tree View, with candidate accuracy shown via overlaid bar charts.
- *Exploratory commands* (e.g., "*Show me essential attributes of a truck*") update the concept visualization panel to show thumbnails of key parts and segment names (Figure 1 D).

To implement supporting Natural Language, we leveraged the ChatGPT API [19], specifically the "gpt-3.5-turbo" model, to implement these features. Detailed instructions are then sent to the ChatGPT API [19]. These prompts break down all possible operations and include examples for each through few-shot prompting, thereby enhancing the model's performance and accuracy in understanding and executing user commands. Each user input is processed through a modular prompt engineering pipeline that identifies the user's intent (e.g., comparison, description, attribute search) and invokes corresponding UI updates, such as modifying the media viewer, highlighting relevant nodes in the tree, or surfacing contrasting examples.

Media Viewer

The Media Viewer serves as the central canvas for all visual feedback. The Media Viewer adapts dynamically to the user's task and interaction context, serving as a flexible visual workspace (*DG3*). For comparison tasks, it presents images side by side to facilitate visual differentiation between instances or classes. It acts as the primary visual ground for contextualizing model predictions, letting users inspect, compare, and annotate visual features with minimal effort. Users can directly interact with visual elements—selecting objects

or specific parts within the image—to generate targeted queries in the Conversational Chat Box (*DG2*).

In *Examples Viewer* mode (Figure 1 A), the Media Viewer displays all instances from the predicted class. This allows users to visually compare the current input to a range of typical examples, helping them assess how well the prediction matches known patterns for that class. When needed users can use *Comparison Viewer* mode (Figure 1 B) to see side-by-side comparisons of the current instance with one or more candidate classes. This enables users to evaluate subtle differences in appearance across categories, supporting fine-grained decisions in ambiguous cases. The visual contrasts help users determine which class the instance more closely resembles.

Moreover, when exploring object segments, users may either click directly on the image to reveal highlighted masks or switch to a *labeled diagram* (Figure 1 E) view that combines annotated segment names with visual overlays (*DG1*). The labeled diagram (Figure 1 E) overlays part names directly on the image, enabling users to visually inspect and verify the correctness of both part segmentation and labeling. This is especially helpful when reasoning about part-based classification systems or when segment-level information is used in prediction. It aids users in spotting mislabelings, missing parts, or inaccurate boundaries, and supports fine-grained correction and feedback.

In addition, the *essential attribute* view (Figure 1 D) highlights key part-level segments that the model has learned to associate with a particular class. These are considered "essential" features for correct classification—for example, a cat's pointed ears or a car's headlights. By surfacing these segments, users can verify whether the model is focusing on semantically meaningful parts and detect cases where it mistakenly treats non-essential objects (like a hair clip in Figure 2 C) as defining features.

Concept Visualization Panel (Radial Tree)

To support exploration of different classes and their hierarchical relationships, the system provides a radial tree visualization at the bottom of the interface. This view presents the full depth of the classification hierarchy, allowing users to navigate and inspect class relationships across multiple levels. Its radial layout gives an overview of the structure before interaction, helping users understand the global organization of categories at a glance. In our model, objects are organized within a hierarchical classification structure, where each label is associated with a confidence score. The tree view presents the model's top classification candidates with accuracy scores at each level dynamically when an example is selected in different views such as example or comparison view (Figure 1 A and B). This feature aligns with *DG3*, as it eliminates the need to ask for confidence levels and labels individually for each class in the hierarchy. Instead, the system dynamically surfaces these explanations as users click on different images, adapting the view based on the context of their interaction. Predicted paths are automatically highlighted, and local actions (e.g., selecting a node) are visually echoed in the media viewer and chat box, reinforcing a sense of coordinated awareness across views. Users can interact with the tree using the same operations available through natural language input. By selecting nodes, they can explore example instances, compare alternatives, and inspect essential attributes and their corresponding labels at different levels of hierarchy.

System Usage

A core strength of MuCHEx lies in its ability to facilitate debugging at various levels of a model's hierarchical classification, moving beyond mere leaf-node errors to address more fundamental confusions. In this section, we illustrate this capability through a walkthrough demonstrating how MuCHEx can be utilized to diagnose and understand higher-level classification errors in hierarchical object classification.

We discuss an example is based on classification of different types of mechanical tools (i.e., hammers, wrenches, screwdrivers), as would be beneficial for guidance assistance for mechanical repair support or when cataloging repair equipment as are included in mechanical tools classification dataset. Imagine a scenario where the model misidentifies a "ball peen hammer", which is a type of "striking tool", as a "socket wrench", which is a type of "wrench". The user, noticing this initial classification error, begins the debugging process by viewing the item within MuCHEx's radial tree to examine the classification labels and their hierarchical relationships (Figure 1 B). To better understand the model's confidence and to explore alternative classifications, the user can ask the system in natural language, "What is this?" The system responds with several candidate labels. While the top candidate shows identification as a "wrench" with medium confidence, the second candidate suggests it may be a "striking tool", specifically a "ball peen hammer". This prompts the user to further explore the explanation behind the misclassification. This confirms the initial *classification error* and motivates the user to investigate the root cause, leveraging MuCHEx's

capabilities to examine the model's reasoning.

To understand why this error occurred, the user's next step is to ask the system, "Why is this a wrench?" In response, the system presents the essential attributes it used to reach that conclusion (Figure 1 C). Here, the user might discover a *segmentation error*. For example, the system may indicate that the presence of "PartX" led it to classify the item as a "wrench", mistakenly identifying "PartX" as an essential attribute of a wrench rather than a striking tool. However, the user knows this part is not essential. This reveals that the system either failed to recognize a key segment or incorrectly marked a non-essential part as critical. To address this, the user can provide the system with more examples—such as a wider variety of striking tools—to help it learn to correctly distinguish and segment "PartX" from truly essential attributes.

Finally, the user may encounter a deeper issue: an *essential attribute* error. Using MuCHEx's labeled diagrams or by clicking on individual segments of the misclassified instance, the user can manually inspect all detected parts (Figure 1 E). This allows them to verify whether the system segmented the object correctly but assigned undue importance to an incorrect part. Upon identifying this, the user can fix the error by teaching the system via natural language—for example, saying, "this is a striking tool, not a wrench". The system learns from this correction. The user can then repeat this debugging process if other levels of the hierarchy remain misclassified, ensuring that both the final classification and the underlying reasoning are accurate and robust.

USER EVALUATION

To evaluate the debugging tool and assess the effectiveness of the different interface components in an exploratory debugging scenario, we conducted a user study. The study was approved by our institution's Institutional Review Board (IRB). For this study, we used a model (described in the "Model" section) that was trained on object classification with images of Oxford-IIIT Pet [17] datasets (cats and dogs with different breeds).

Study Design and Procedure

We conducted a lab-based user study to evaluate how participants interact with our system in a few-shot classification setting (4-shot learning). We selected a 4-shot learning setting because, through pilot testing, we found that it reflects a diverse range of model failures across different levels of the class hierarchy, including misclassifications, segmentation issues, and incorrect attribute associations. This makes it well-suited for evaluating our system's ability to support error analysis. At the same time, limiting each class to four instances keeps the total dataset size manageable for participants, avoiding cognitive overload and keeping the study duration within reasonable bounds. Importantly, this aligns with real-world challenging scenarios where labeled data is scarce, and effective debugging becomes especially critical to ensure model reliability.

Also, the goal of the study was to evaluate the effectiveness of our tool in helping users identify different types of errors, uncover interaction patterns, and explore how various features were used in alignment with our design goals. To systematically evaluate the model errors identified in our earlier analysis, we designed three targeted task types, each corresponding to one of the main error categories. Specifically, participants were asked to complete three tasks: (1) identify and correct misclassified objects, providing a reason for each correction based on the model's output (See Figure 2 A); (2) find an instance of a breed with incorrect or nonsensical segmentation (See Figure 2 B); and (3) detect an incorrectly marked essential attribute of a breed—i.e., one that was classified as essential but should not be (See Figure 2 C).

We used the Oxford-IIIT Pet dataset [17], which contains images of cats and dogs across various breeds. For the study, we selected a random subset of 12 classes due to time constraints of studies. The model was trained using 4-shot learning to simulate limited data conditions, and its classification accuracy across different levels is reported in Table 1. This setup allowed us to test the tool in a challenging but realistic scenario where users could explore the limits of model understanding and engage with different levels of classification and annotation errors.

After completing the tasks, participants filled out a questionnaire rating their usage of and the perceived helpfulness of each interface element on an ordinal scale. Each session lasted approximately 60 minutes, allowing participants to proceed at their own pace. We collected participants' responses to the main debugging task, along with comprehensive interaction logs. To evaluate users' mental model of our system, participants also completed a post-study questionnaire. Participants' satisfaction with the explanations and their trust in the system were both assessed using scales adapted from Hoffman et al. [20], specifically the *Explanation Satisfaction Scale* and the *Trust Scale for the XAI context*. This was followed by a short informal interview to understand their strategies and experiences better. We recruited 10 participants (8

males and 2 females), all undergraduate or graduate students in computing-related fields, as we sought participants with familiarity with machine learning for the debugging context of the study.

Study Results

Interaction Patterns To understand how participants interacted with the system across modalities and views, we analyzed their usage patterns during the task. We observed that the majority of interactions (80.35% average per participant) were conducted through GUI-based inputs, while 19.65% involved natural language (NL) input. This interaction ratio aligns with *DG2*, which supports flexible, dual-modal interaction. Participants frequently used NL input to initiate exploration or inquire about high-level information—for example, asking for explanations, querying essential attributes, or requesting class comparisons. These natural language interactions often served as entry points into the task, after which participants relied more heavily on GUI components to drill down into finer-grained inspection, such as examining segmentations or navigating the class hierarchy for detailed comparisons. Additionally, it was notable that 70% of all NL inputs were issued while participants had an image or part of an image in focus (e.g., while viewing an object in the Media Viewer or referencing a labeled segment using "this"). This shows the different use cases for modalities: users relied on language to ask further details when referring to an object and on GUI actions for clear goal-driven tasks.

Additionally, we calculated the average percentage of how frequently participants interacted with each view. The *Concept Visualization Panel (Tree)* was used most frequently, averaging 53.62% of interactions, followed by the *Media Viewer* at 26.73%, and the *Conversational Chat Box* at 19.65%. These results support *DG1*, which emphasizes the need for multi-level exploration across both instance-level visual evidence and higher-level class hierarchies. The greater use of the Tree View suggests that participants frequently navigated and inspected relationships between categories, while the Media Viewer supported image-based reasoning, and the Chat Box enabled clarification or exploratory questions. Together, these patterns demonstrate that participants made use of all three coordinated views in complementary ways: validating *DG1* for enabling cross-level inspection, and *DG2* for supporting fluid transitions between GUI interactions and natural language inputs based on the nature of the task.

A closer examination of the natural language (NL) inputs revealed that 39 inputs (16% of all NL inputs) were used to trigger features such as the *Examples View* (Figure 1 A) and *Comparison View* (Figure 1 B), which typically require a sequence of at least three GUI-based actions (e.g., selecting a node, navigating to the comparison menu, and choosing a reference point). Among these, 32 commands invoked comparisons and 7 requested examples. This behavior indicates that participants turned to NL input when the required GUI interactions became too complex or cumbersome, highlighting its value for accessing advanced functionality with minimal effort. However, when using direct manipulation, participants manually opened the *Comparison Viewer* 112 times and the *Examples Viewer* 205 times. These figures suggest that while GUI interactions were used frequently, NL served as a strategic shortcut when tasks involved more depth or coordination across multiple elements. Together, these findings reinforce the value of enabling *DG2* for balancing efficiency and control, particularly in complex debugging scenarios.

Moreover, to provide some quantitative data for the complexity of the debugging tasks, we logged participants' completion times and their responses. The tasks varied in duration, reflecting their different challenges. Task 1, correcting misclassifications, was the most involved (*Mean* = 15.76 min, *Median* = 18.19 min, *Range* = [5.79–20.51 min]). In contrast, Task 2, which required finding segmentation errors, and Task 3, focused on detecting incorrect essential attributes, demanded less time (Task 2 *Mean* = 5.79 min, *Median* = 4.21 min; Task 3 *Mean* = 6.00 min, *Median* = 4.34 min). Upon manually evaluating each participant's answers, we found that all 10 participants successfully accomplished three tasks (100% task success).

Mental Model Questionnaire The results of the post-task explanation and trust questionnaire, adapted from Hoffman et al. [20], indicate generally strong agreement about the system's efficiency and usefulness in helping users understand how the tool works, as shown in Table 2. Notably, items related to explanation satisfaction, sufficiency, and predictability received high agreement scores ($\geq$60%).

Consider the lower agreement scores (60% for "*The outputs of the tool are very predictable.*" and "*This explanation of how the tool works seems complete.*") Reduced predictability is a natural side effect of adaptive or dynamic interfaces [21]. While this adaptivity can lead to perceived unpredictability, it offers significant benefits such as reducing cognitive effort and the steps needed to find the details for the task. It may also stem from the debugging context, as participants were

This article has been accepted for publication in IEEE Computer Graphics and Applications. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/MCG.2025.3598204

TABLE 2. Post task questions (adapted from Hoffman et al. [20]) for which at least 60 % of participants selected "agree" or "strongly agree".

Question text	*Agreement*
From the explanation, I know how the tool works.	100%
This explanation of how the tool works is satisfying.	100%
This explanation of how the tool works tells me how to use it.	90%
This explanation of how the tool works is useful to my goals.	90%
The tool is efficient in that it works very quickly.	80%
This explanation of how the tool works has sufficient detail.	80%
The outputs of the tool are very predictable.	60%
This explanation of how the tool works seems complete.	60%
This explanation of the tool shows me how accurate the tool is.	60%

explicitly tasked with exploring error cases. This likely exposed them to more unexpected outputs, reducing their ability to reliably anticipate the system's behavior.

Although three items specifically related to trust in the system's outputs received lower agreement, this aligns with the experimental setup: participants were tasked with identifying and correcting system errors. As such, their lower trust ratings reflect appropriate skepticism and suggest that participants were correctly engaged in a critical evaluation of the tool's behavior.

Post-Task Interview Following the task, we conducted semi-structured interviews with participants to gain qualitative details about their interaction experiences. Participants were asked open-ended questions such as "What features did you use the most?", "Which ones helped you make decisions or spot issues?", and their general thoughts on the use of multimodal input and multiple coordinated views.

All participants emphasized the tool's capability to identify diverse errors occurring at various levels, as well as pointed out several features they found especially beneficial. The *Labeled Diagram* (Figure 1 E) view was the most frequently cited, with 8 out of 10 participants noting it was essential for identifying incorrect segmentations. Majority of participants (7 out of 10) appreciated the *flexibility of interaction*, stating they alternated between GUI-based interaction (e.g., Tree View) and natural language input depending on the nature of the task. Also, two participants specifically noted that they preferred GUI-based interaction using the *tree view* when there was a clear context or goal, but found natural language input more helpful when they were unsure how to begin or needed more details. In addition, three participants specifically highlighted the *Comparison* (Figure 1 B) feature as particularly useful for confirming that an instance was assigned to the correct class. Similarly, three participants pointed to the *Essential Attributes* (Figure 1 D) view as useful for understanding the reasoning behind the system's classifications.

DISCUSSION

In this paper, we introduced a multimodal visual tool that provides multi-level explanations to support users in understanding and debugging different types of classification errors. The system was grounded in three user-centered design goals focused on supporting hierarchical navigation, multimodal interaction, and adaptive information presentation. To evaluate the system, we tested it on three different datasets (Table 1). Our findings highlight that while object classification models perform confidently on high-level classes—such as distinguishing between an animal and a vehicle—they struggle with fine-grained classification tasks, such as identifying specific breeds or models, especially under limited data conditions (e.g., few-shot learning). More broadly, this work illustrates the value of combining multiple modes of interaction, such as visual and natural language, for supporting diverse user goals in model debugging. It also highlights the importance of dynamically adapting explanations to the user's current task and input type, rather than relying on static or one-size-fits-all feedback. This provides concrete evidence that flexible, coordinated interaction modalities and context-aware explanation delivery can enhance usability and interpretability in complex decision spaces.

To assess the effectiveness of our tool in this challenging setting, we trained a model using 4-shot learning on the Oxford-IIIT Pet dataset [17] and conducted a lab-based user study with 10 participants. Participants were tasked with identifying and correcting three types of errors, such as high-level misclassification, incorrect reasoning, and segmentations. This task was intentionally difficult, as the model achieved only 52.5% accuracy in breed-level classification, compared to 99.1% accuracy in identifying the broader animal category (Table 1). Despite this, all 10 participants successfully completed all three tasks, demonstrating the tool's effectiveness in supporting error discovery across multiple classification levels. Interaction log

analysis further supports the utility of our design goals. As detailed in the *Study Results* Section, all three coordinated views—the concept tree, media viewer, and NL interface—were frequently used, validating *DG1*. Participants explored both hierarchical structures and instance-level details using these synchronized views. In line with *DG2*, while the majority of interactions (80.35%) were performed through GUI-based actions, natural language input accounted for 19.65% of interactions. Notably, most NL queries were issued in reference to specific images or segments, suggesting that users relied on language to request additional contextual explanations, while GUI interactions were preferred for direct, goal-oriented tasks. Overall, these findings demonstrate that integrating multimodal interaction and coordinated views can significantly aid users in navigating complex hierarchical classification systems and identifying subtle model failures.

Moreover, the ability to explore classification decisions through our conversational visual tool provides opportunities for understanding model behavior than single post hoc explanations alone. Our task-driven, adaptive interface design contributes to the growing literature in interactive explanation such as supporting retrospective analysis [22] key goals that are difficult to meet with static explanation systems.

Beyond visual inspection, our system highlights the value of natural language (NL) as a flexible and expressive modality for interacting with machine learning models. In our user testing, participants strategically used NL to initiate comparisons, inspect alternatives, and request reasoning tied to visual evidence. Existing systems (e.g., [23], [24]) have demonstrated that the integration of NL into visual tools can allow for a more natural and accessible inquiry process. In addition to broadening interaction modalities, there is also an opportunity to expand the types of explanations supported by the system. While our current system supports NL explanations tied to visual outputs, future work could expand the scope of explanation types.

Limitations and Future Work

While our design effectively supports few-shot learning scenarios where users need to debug classifications over a small number of instances and classes, this focus also introduces scalability limitations. In our evaluation and design process, we primarily targeted use cases where the model has only seen a few examples per class. Accordingly, visualizations such as the radial tree and image grid were optimized for small-scale exploration. However, these techniques may become cluttered or unwieldy as the number of classes or instances increases, for example, in real-world scenarios involving hundreds of categories or thousands of predictions. This limitation suggests the need for more scalable visual paradigms. Future work should explore alternative visualization strategies, such as collapsible tree structures, zoomable node-link diagrams, or cluster-based summarizations to better support large hierarchical datasets. Additionally, features such as search and filtering across the class hierarchy could ease navigation and reduce visual overload.

Another limitation of our current system is its focus on instance-by-instance analysis. While this mode is well-suited for understanding localized errors in few-shot learning settings, it is insufficient for debugging cases where analysts need to identify patterns across many samples. For example, users may want to detect systematic errors, such as recurring misclassifications between similar classes. To support this, future iterations of the system should enable batch analysis capabilities, for example, tools to "show all instances where the model confused class A with class B" or to explore clusters of misclassified examples. Supporting aggregate-level alongside single-instance views would significantly enhance the tool's usefulness in large-scale debugging tasks. Finally, our evaluation with the Oxford-IIIT Pet dataset [17] focused on leaf-node classifications. This is the case where models face the most challenging distinctions, such as distinguishing between African elephants and Asian elephants, or even more specific contrasts like African savannah and African forest elephants, which can be challenging even for humans. However, since this dataset's top-level accuracy of over 99% for top-level classification (e.g., distinguishing animals), we also added a use case example in "*System Usage*" to explicitly showcase how MuCHEx would be used to diagnose a higher-level error, such as a model confusing a coyote with a wolf.

CONCLUSION

This paper contributes a novel multimodal conversational debugging tool, MuCHEx, that integrates coordinated views, natural language interaction, and adaptive explanations for hierarchical object classification. We also present findings from a human-subjects study demonstrating the tool's effectiveness in helping users identify classification errors. Beyond the scope of this study, our tool can be serve as a general-purpose debugging assistant for object classification tasks. Before deploying models into real-world, domain-specific applications—such as medical imaging, autonomous driving—developers can use this system to explore

failure cases, verify confidence levels, and trace reasoning pathways across the model's hierarchy. For example, developers of autonomous driving models could use the system to distinguish whether misclassifications occur between nearby classes (e.g., pedestrian vs. bicyclist) or stem from deeper issues in the model's visual segmentation. Such pre-deployment debugging workflows are especially crucial in high-stakes or low-data domains. Our system enables developers and non-experts alike to collaboratively investigate and refine model predictions, making it a valuable tool in pipelines. Future work may explore integration with active learning or automated retraining pipelines, leveraging user-identified errors to improve model performance iteratively.

ACKNOWLEDGMENTS

This work was supported by the DARPA ECOLE Program under award number HR00112390063.

REFERENCES

1. X. Zhang, Y.-H. Yang, Z. Han, H. Wang, and C. Gao, "Object class detection: A survey," *ACM Computing Surveys (CSUR)*, vol. 46, no. 1, pp. 1–53, 2013.
2. B. Zhao, J. Feng, X. Wu, and S. Yan, "A survey on deep learning-based fine-grained object classification and semantic segmentation," *International Journal of Automation and Computing*, vol. 14, no. 2, pp. 119–135, 2017.
3. M. A. Grasso, D. S. Ebert, and T. W. Finin, "The integrality of speech in multimodal interfaces," *ACM Transactions on Computer-Human Interaction (TOCHI)*, vol. 5, no. 4, pp. 303–325, 1998.
4. S. Mohseni, N. Zarei, and E. D. Ragan, "A multidisciplinary survey and framework for design and evaluation of explainable ai systems," *ACM Transactions on Interactive Intelligent Systems (TiiS)*, vol. 11, no. 3-4, pp. 1–45, 2021.
5. M. Q. Wang Baldonado, A. Woodruff, and A. Kuchinsky, "Guidelines for using multiple views in information visualization," in *Proceedings of the working conference on Advanced visual interfaces*, 2000, pp. 110–119.
6. A. Saktheeswaran, A. Srinivasan, and J. Stasko, "Touch? speech? or touch and speech? investigating multimodal interaction for visual network exploration and analysis," *IEEE transactions on visualization and computer graphics*, vol. 26, no. 6, pp. 2168–2179, 2020.
7. B. Steichen, G. Carenini, and C. Conati, "User-adaptive information visualization: using eye gaze data to infer visualization tasks and user cognitive abilities," in *Proceedings of the 2013 international conference on Intelligent user interfaces*, 2013, pp. 317–328.
8. M. D. Rahman, B. Doppalapudi, G. J. Quadri, and P. Rosen, "A survey on annotations in information visualization: Empirical insights, applications, and challenges," *arXiv preprint arXiv:2410.05579*, 2024.
9. P. W. Koh, T. Nguyen, Y. S. Tang, S. Mussmann, E. Pierson, B. Kim, and P. Liang, "Concept bottleneck models," 2020. [Online]. Available: https://arxiv.org/abs/2007.04612
10. A. K. Menon, A. Rajagopalan, B. Sumengen, G. Citovsky, Q. Cao, and S. Kumar, "Online hierarchical clustering approximations," 2019. [Online]. Available: https://arxiv.org/abs/1909.09667
11. M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, M. Assran, N. Ballas, W. Galuba, R. Howes, P.-Y. Huang, S.-W. Li, I. Misra, M. Rabbat, V. Sharma, G. Synnaeve, H. Xu, H. Jegou, J. Mairal, P. Labatut, A. Joulin, and P. Bojanowski, "Dinov2: Learning robust visual features without supervision," 2024. [Online]. Available: https://arxiv.org/abs/2304.07193
12. D. Srivastava, G. Yan, and L. Weng, "VLG-CBM: training concept bottleneck models with vision-language guidance," in *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, A. Globersons, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. M. Tomczak, and C. Zhang, Eds., 2024. [Online]. Available: http://papers.nips.cc/paper_files/paper/2024/hash/90043ebd68500f9efe84fedf860a64f3-Abstract-Conference.html
13. M. Yüksekgönül, M. Wang, and J. Zou, "Post-hoc concept bottleneck models," in *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. [Online]. Available: https://openreview.net/forum?id=nA5AZ8CEyow
14. H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 67, no. 2, pp. 301–320, 2005. [Online]. Available: https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-9868.2005.00503.x
15. J. Krause, M. Stark, J. Deng, and L. Fei-Fei, "3d object representations for fine-grained categorization," in *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, Sydney, Australia,

2013.

16. S. Maji, J. Kannala, E. Rahtu, M. Blaschko, and A. Vedaldi, "Fine-grained visual classification of aircraft," Tech. Rep., 2013.
17. O. M. Parkhi, A. Vedaldi, A. Zisserman, and C. V. Jawahar, "Cats and dogs," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.
18. F. M. Miranda, N. Köhnecke, and B. Y. Renard, "Hiclass: a python library for local hierarchical classification compatible with scikit-learn," *J. Mach. Learn. Res.*, vol. 24, pp. 29:1–29:17, 2023. [Online]. Available: https://jmlr.org/papers/v24/21-1518.html
19. OpenAI, "Chatgpt," 2024, large language model. [Online]. Available: https://www.openai.com
20. R. R. Hoffman, S. T. Mueller, G. Klein, and J. Litman, "Measures for explainable ai: Explanation goodness, user satisfaction, mental models, curiosity, trust, and human-ai performance," *Frontiers in Computer Science*, vol. 5, p. 1096257, 2023.
21. K. Z. Gajos, M. Czerwinski, D. S. Tan, and D. S. Weld, "Exploring the design space for adaptive graphical user interfaces," in *Proceedings of the working conference on Advanced visual interfaces*, 2006, pp. 201–208.
22. B. Shneiderman, *Human-centered AI*. Oxford University Press, 2022.
23. Y. Guo, D. Shi, M. Guo, Y. Wu, N. Cao, and Q. Chen, "Talk2data: A natural language interface for exploratory visual analysis via question decomposition," *ACM Transactions on Interactive Intelligent Systems*, vol. 14, no. 2, pp. 1–24, 2024.
24. B. Yu and C. T. Silva, "Flowsense: A natural language interface for visual data exploration within a dataflow system," *IEEE transactions on visualization and computer graphics*, vol. 26, no. 1, pp. 1–11, 2019.